UNIVERSITÄT BONN

Efficient Perception and Forecasting for Autonomous Vehicles

DISSERTATION

zur Erlangung des Doktorgrades (*Dr. rer. nat.*)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität, Bonn

vorgelegt von

SHIJIE LI

aus

Hebei, China

Bonn, 2024

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn

1. Gutachter / 1st Advisor: Prof. Dr. Juergen Gall
2. Gutachter / 2nd Advisor: Prof. Dr. Abhinav Valada
Tag der Promotion / Day of Promotion: 19.04.2024
Erscheinungsjahr / Year of Publication: 2024

Abstract

by Shijie Li

for the degree of

Doctor rerum naturalium

In recent years, autonomous driving vehicles have attracted a lot of attention. In autonomous driving, safety is the highest priority. To ensure this, scene understanding and motion forecasting play an important role in autonomous driving. Because the LiDAR sensor can capture environment information accurately, in the scene understanding task, we want to assign each point a semantic label. Unfortunately, there are usually massive points lying in the LiDAR point cloud. Processing such a large point cloud usually requires a lot of computation resources. However, only limited computation resources are available on autonomous driving platforms. What's worse, there are usually many modules that run concurrently on autonomous driving which further limits the available computation. This is the same for motion forecasting tasks. Thus we can observe that efficiency is very important for realistic autonomous driving applications.

However, previous methods targeting the semantic segmentation of LiDAR point clouds have largely focused on accuracy, often at the expense of efficiency, which is crucial for practical applications. To bridge this gap, we propose a highly efficient architecture tailored to process 2D projection maps generated from input LiDAR sensors. By capitalizing on the intrinsic characteristics of LiDAR sensors and converting sparse 3D data into a dense 2D format, our methodology achieves a commendable balance between accuracy and efficiency. Building on this architecture, we further endeavor to determine the motion status of each point within the LiDAR point cloud. This is realized by adopting a generalized projection-based LiDAR data representation that can encapsulate motion information. Additionally, we have set a new benchmark for LiDAR-based moving object segmentation, anchored on the SemanticKITTI dataset. The aforementioned discussions primarily revolve around projection-based approaches. Moreover, we explore the enhancement of point-based methods for semantic segmentation of LiDAR point clouds. We present a novel concept of reconfiguring 3D point-based operations to operate within the projection space. This ingenious design allows for the conversion of any point-based methods into projection-based ones, markedly improving their accuracy and efficiency for LiDAR point cloud semantic segmentation. A significant hurdle with projection-based methods is the apparent gap between them and voxel-based approaches. We ascertain that this gap arises from the "many-to-one" dilemma, due to the limited horizontal and vertical angular resolution of the range image. To tackle this, we introduce a temporal fusion layer to extract pertinent information from previous scans, integrating it with the current scan. We also suggest a max-voting-based post-processing technique to amend erroneous predictions.

As highlighted, besides scene understanding, precise and efficient trajectory forecasting is paramount for ensuring safety in autonomous driving. As the safety of humans is the highest priority, we first work on mapless human trajectory prediction. In this work, we en-

force the temporal consistency between historical data and predicted future data. Although this method works well for low-speed scenarios that mainly consist of pedestrians, the autonomous driving scenario is usually highly dynamic where the agents are usually at high speed, like vehicles. In this case, the map can provide meaningful information as the drivers usually need to follow some driving rules. One issue here is that processing map information usually costs a lot of computation resources. To address this, we propose a streamlined architecture that incorporates an additional aggregation token for each lane and trajectory. This facilitates the modeling of global dependencies within each lane or trajectory and between the lanes or trajectories, leading us to name the network the Aggregation-Interaction Transformer. Nonetheless, efficiently modeling the interactions among various road users and drivable lanes remains a formidable challenge. Thus another notable contribution is our approach's ability to learn intention seeds, which serve as queries to generate a diverse array of future trajectories in a highly efficient manner.

For both the LiDAR point cloud semantic segmentation and motion forecasting tasks, we evaluate the proposed approaches on several public datasets with different scenarios. Extensive evaluation shows the effectiveness of the proposed approaches in both accuracy and efficiency which shows they are suitable for realistic applications.

Keywords: LiDAR semantic segmentation, Motion forecasting, Autonomous driving, Scene understanding

Acknowledgements

I would like to thank my advisor Prof. Juergen Gall for all the guidance and support. I was privileged during my PhD and had the freedom to work on my own ideas and yet he always provided thorough feedback and many suggestions to polish these ideas and improve the quality of the work. I have been always amazed by the amount of effort and time he puts in until the last moment before the deadlines to refine our submissions.

Furthermore, I would like to thank the dissertation committee members: Prof. Abhinav Valada, Prof. Maren Bennewitz, and Prof. Sebastian Neubert for taking the time to review the thesis.

I'm also grateful to my colleagues and friends in the Computer Vision Group at the University of Bonn. Their support, kindness, and insightful discussions made my stay at Bonn a pleasant experience. I would like also to thank my collaborators for the fruitful collaboration.

Finally, many thanks to my parents, and wife for their unconditional love and support.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in the thesis:

BPTT	Back-Propagation Through Time
CNN	Convolutional Neural Network
DDL	Dual Dilated Layer
EOS	End-Of-Sequence
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HOF	Histograms of Optical Flow
HOG	Histograms of Oriented Gradients
IDT	Improved Dense Trajectories
IoU	Intersection over Union
LSTM	Long Short-Term Memory
MBH	Motion Boundary Histograms
MLP	Multi-Layer perceptron
MoC	Mean over Classes
MSE	Mean Squared Error
MS-TCN	Multi-Stage Temporal Convolutional Network
PCA	Principal Component Analysis
RBF	Radial Basis Function
RNN	Recurrent Neural Network
SOS	Start-Of-Sequence
SS-TCN	Single-Stage Temporal Convolutional Network
SVM	Support Vector Machine
TCN	Temporal Convolutional Network
TN	True Negative
TP	True Positive
TSN	Temporal Segment Network

List of Publications

The thesis is based on the following publications:

- **Multi-scale Interaction for Real-time LiDAR Data Segmentation on an Embedded Platform**
  ShiJie Li, Xieyuanli Chen, Yun Liu, Dengxin Dai, Cyrill Stachniss, Juergen Gall
  DOI: 10.1109/LRA.2021.3132059
  IEEE Robotics and Automation Letters (RA-L), 2021.

- **Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data**
  Xieyuanli Chen, Shijie Li, Benedikt Mersch, Louis Wiesmann, Juergen Gall, Jens Behley, Cyrill Stachniss
  DOI: 10.1109/LRA.2021.3093567
  IEEE Robotics and Automation Letters (RA-L), 2021.

- **Rethinking 3D LiDAR Point Cloud Segmentation**
  ShiJie Li, Yun Liu, Juergen Gall
  DOI: 10.1109/TNNLS.2021.3132836
  IEEE Transactions on Neural Networks and Learning Systems, 2021.

- **Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting**
  ShiJie Li, Yanying Zhou, Jinhui Yi, Juergen Gall
  DOI: 10.1109/ICCV48922.2021.00195
  International Conference on Computer Vision, 2021.

CHAPTER 1

Introduction

Contents

1.1 Motivation

Autonomous driving, also known as self-driving or driverless driving, refers to the technology that allows a vehicle to navigate and operate without any human intervention or control. This technology leverages a variety of systems and tools to function properly.

The autonomous driving system usually consists of several modules:

- Perception system: This is the initial step in autonomous driving where the vehicle collects data from the surrounding environment. To ensure safe autonomous driving, various sensors are used including LiDAR, radar, camera, IMU, and so on.
- Scene Understanding: The system interprets the data to understand the driving scene. It usually involves object detection, object tracking, and semantic segmentation.
- Anticipation: In this phase, the vehicle anticipates the actions of other road users and makes decisions accordingly. As there are usually many potential trajectories, this module usually predicts diverse outcomes.

- Control: The control module in autonomous driving is a critical system responsible for executing the decisions made by the planning module to maneuver the vehicle safely and efficiently.

Although vehicles equipped for autonomous driving incorporate a range of sensors, the LiDAR sensor stands as the most critical one. LiDAR, which stands for Light Detection and Ranging, is a sophisticated remote sensing technology that utilizes pulsed laser light to gauge distances. This technology is capable of generating detailed 3D maps of the surroundings, capturing the intricate details of various objects, and offering a 360-degree field of view, thereby ensuring a comprehensive understanding of the environment. LiDAR excels in identifying obstacles with a high degree of precision, including pedestrians and cyclists, thereby significantly enhancing safety measures. One of its standout features is its independence from ambient light, which facilitates effective functioning in low-light conditions, a capability not shared by cameras. Furthermore, LiDAR maintains its functionality across a diverse range of weather conditions, including during rain and fog, conditions that often challenge other sensors such as cameras. When used in harmony with other sensors, LiDAR contributes to a redundancy system, bolstering the reliability and robustness of the autonomous driving system.

However, LiDAR sensors typically record the surrounding environment as a large-scale point cloud composed of a vast number of points. Analyzing such extensive point cloud data demands substantial computational resources. Regrettably, autonomous vehicles often rely on embedded platforms, which offer limited computational capabilities. Compounding this issue is the fact that these platforms frequently run multiple modules simultaneously, as previously noted. This concurrent operation further restricts the computational resources available for processing LiDAR data in scene understanding applications. Therefore, it is evident that efficiency is crucial when it comes to processing LiDAR data in autonomous driving contexts. Regrettably, the inherent characteristics of LiDAR point clouds, which are large-scale yet sparse, present unique challenges. Many existing methods are not designed to handle such large-scale sparse point clouds effectively, often failing to strike a harmonious balance between accuracy and efficiency.

In the realm of autonomous driving, the anticipation module operates synergistically with the scene understanding module to enhance safety through meticulous trajectory forecasting for each agent present in the prevailing scenario. This module is tasked with predicting the forthcoming trajectories of all entities on the road - vehicles, pedestrians, or cyclists- leveraging both their current states and historical data. Functioning in real-time, it continually refreshes its predictions, drawing from the most recent data harvested from the vehicle's sensor suite. Through the utilization of cutting-edge technologies and sophisticated algorithms, the module scrutinizes the movements and behavioral patterns of various road users to forecast their future paths. This predictive prowess stands as a fundamental component in the autonomous vehicle's toolkit, facilitating safe and efficient navigation through ever-changing environments, thereby cementing itself as a cornerstone in the technology driving autonomous vehicles.

Moreover, it is imperative to comprehend and model the intricate social interactions between road users, a task steeped in complexity given the unpredictable nature of human behavior and the diverse range of scenarios that can unfold on the roads. Adding to the complexity is the dynamic environment in which autonomous vehicles operate, requiring real-time trajectory forecasting to respond swiftly to evolving situations. This demands high computational resources, which are often limited, thereby restricting the complexity of forecasting models that can be deployed. Therefore, there is a pressing need to develop forecasting models that are both sophisticated in understanding

and predicting a range of human behaviors and social interactions, while also being computationally efficient to operate in real-time in a highly dynamic environment.

In this thesis, we present significant contributions to the autonomous driving domain, specifically focusing on enhancing the scene understanding and anticipation modules through LiDAR point cloud semantic segmentation and motion forecasting, respectively.

In the context of scene understanding, we delve into a detailed analysis of various semantic segmentation methods applicable to LiDAR point cloud data. Our discussion elucidates the relationships among these methods, leading to the proposition of techniques that stand out as both highly accurate and efficient, thereby making them apt for deployment in autonomous driving scenarios.

Turning our attention to the anticipation module, we work on both mapless and map-assist trajectory prediction. We first propose a method for mapless method targeting human trajectory prediction which enforces the temporal consistency between historical data and the predicted data. While this method already works well for low-speed scenarios, it is not suitable for autonomous driving scenarios which a highly dynamic scenarios where agents are usually at high speed. Furthermore, the map can provide meaningful information in this case as the cars need to follow some driving rules. Thus we introduce a novel method that can process input maps efficiently, a trait that renders it highly suitable for real-world autonomous driving applications, aligning perfectly with the demands of realistic and dynamic driving environments. It can also delineate agent intentions, effectively capturing the multi-modal characteristics inherent in driving scenarios.

1.2 Challenges

Autonomous driving technology is advancing rapidly, but it still faces a myriad of challenges that need to be addressed to ensure safety, efficiency, and widespread adoption. Here are some of the significant challenges:

1.2.1 Sparse High Dimension data

LiDAR point clouds are inherently sparse compared to image data, making it challenging to accurately identify and segment objects, particularly those that are small or distant. Additionally, LiDAR data exists in a 3D space, escalating the data complexity and the computational resources necessary for processing. This high dimensionality further complicates data storage and transmission, aspects critical for real-time applications in autonomous driving.

1.2.2 Real-Time Processing

As previously noted, processing LiDAR point cloud data typically demands substantial computational resources. However, autonomous vehicles operate under constraints concerning computational resources, power consumption, and space. Navigating these constraints while striving for real-time processing of LiDAR data for semantic segmentation poses a complex challenge.

1.2.3 Multi-Modal Predictions in motion forecasting

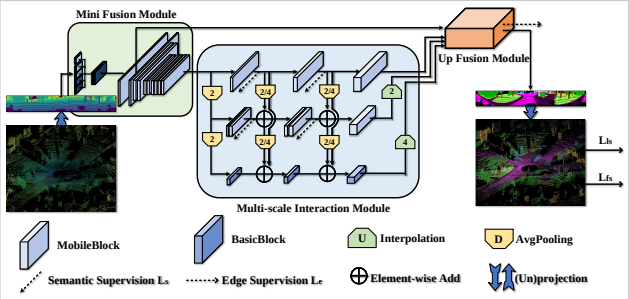
Forecasting the future movements of surrounding entities carries inherent uncertainty. Drivers, pedestrians, and other road users may alter their behavior due to a myriad of unpredictable factors. More-

over, there often exist multiple plausible future trajectories for an object. For example, a vehicle approaching an intersection may turn left, right, or continue straight. Capturing these multi-modal predictions poses a significant challenge.

1.3 Contributions

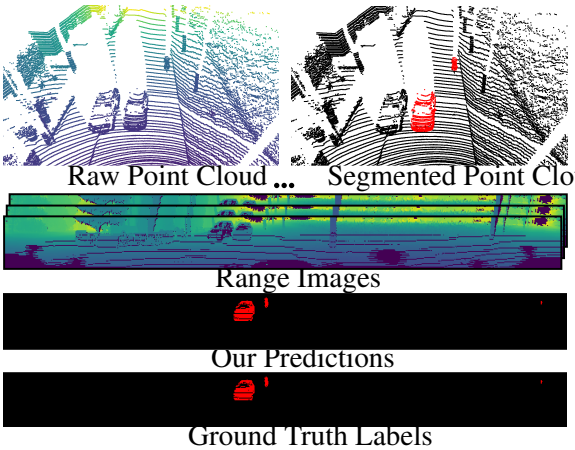
1.3.1 Multi-path architecture for efficient LiDAR point cloud semantic segmentation

Our first contribution is that we propose a projection-based method, called Multi-scale Interaction Network (MINet), which is very efficient and accurate. The network uses multiple paths with different scales and balances the computational resources between the scales. Additional dense interactions between the scales avoid redundant computations and make the network highly efficient. The proposed network outperforms point-based, image-based, and projection-based methods in terms of accuracy, number of parameters, and runtime. Moreover, the network processes more than 24 scans, captured by a high-resolution LiDAR sensor with 64 beams, per second on an embedded platform, which is higher than the framerate of the sensor. The network is therefore suitable for robotics applications.



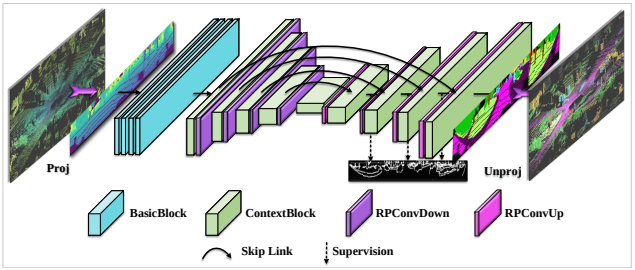
1.3.2 Reducing the influence of moving objects in autonomous driving by moving object segmentation

Apart from semantic information, moving information is also very important in autonomous driving perception modules which is usually ignored in previous research. The ability to detect and segment moving objects in a scene is essential for building consistent maps, making future state predictions, avoiding collisions, and planning. Our second contribution, instead of assigning each point a semantic label, we want to distinguish the moving status of each point. We propose a novel approach that pushes the current state of the art in LiDAR-only moving object segmentation forward to provide relevant information for autonomous robots and other vehicles. Instead of segmenting the point cloud semantically, *i.e.*, predicting the semantic classes such as vehicles, pedestrians, buildings, roads, etc., our approach accurately segments the scene into moving and static objects, *i.e.*, distinguishing between moving cars



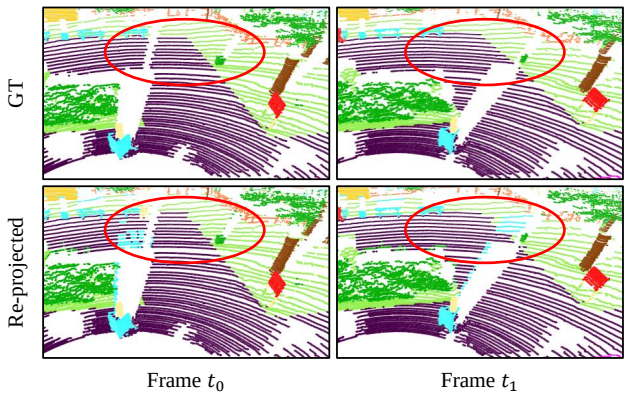
vs. parked cars. Our proposed approach exploits sequential range images from a rotating 3D LiDAR sensor as an intermediate representation combined with a convolutional neural network and runs faster than the frame rate of the sensor. We compare our approach to several other state-of-the-art methods showing superior segmentation quality in urban environments. Additionally, we created a new benchmark for LiDAR-based moving object segmentation based on SemanticKITTI.

1.3.3 Efficient Point-based LiDAR point cloud semantic segmentation by operation projection



Our contributions so far only focus on projection-based methods. Apart from projection-based methods, point-based methods, which directly operate on the raw point cloud, are also widely used in 3D scene understanding. However, most of them have been designed for indoor scenarios, but they struggle if they are applied to point clouds that are captured by a LiDAR sensor in an outdoor environment. In order to make these methods more efficient and robust such that they can handle LiDAR data, we introduce the general concept of reformulating 3D point-based operations such that they can operate in the projection space. While we show by means of three point-based methods that the reformulated versions are between 300 and 400 times faster and achieve a higher accuracy, we furthermore demonstrate that the concept of reformulating 3D point-based operations allows to design of new architectures that unify the benefits of point-based and image-based methods. As an example, we introduce a network that integrates reformulated 3D point-based operations into a 2D encoder-decoder architecture that fuses the information from different 2D scales. We evaluate the approach on four challenging datasets for semantic LiDAR point cloud segmentation and show that leveraging reformulated 3D point-based operations with 2D image-based operations achieves very good results for all four datasets.

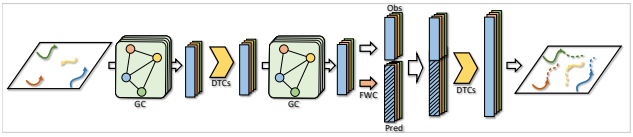
1.3.4 Enforce Temporal consistency in projection-based methods



One issue with the projection-based method is that it faces a significant challenge known as the “many-to-one” problem caused by the range image’s limited horizontal and vertical angular resolution. As a result, around 20% of the 3D points can be occluded. In this contribution, we present TFNet, a range-image-based LiDAR semantic segmentation method that utilizes temporal information to address this issue. Specifically, we incorporate a temporal fusion layer to extract useful information from previous scans and integrate it with the current scan. We then design

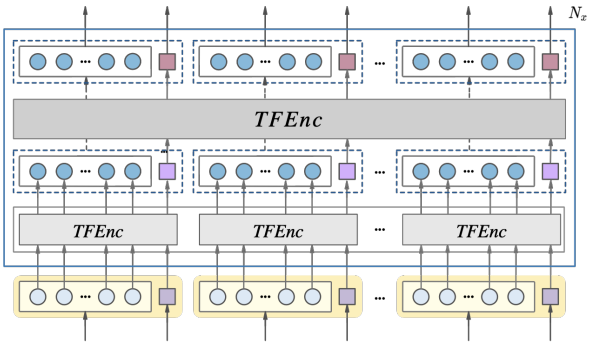
a max-voting-based post-processing technique to correct false predictions, particularly those caused by the "many-to-one" issue. We evaluated the approach on two benchmarks and demonstrated that the post-processing technique is generic and can be applied to various networks.

1.3.5 Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting



Based on the results from the scene understanding module, we want to go one step further and forecast the future trajectories of each agent. In this work, we focus on mapless human trajectory prediction as the safety of humans is of the highest priority. To this end, agents are required to understand and forecast the trajectories of surrounding pedestrians or vehicles such that they can make the right decisions and for instance navigate safely and smoothly through a crowd. Trajectory forecasting, however, is challenging due to the complex behavior and interactions of humans in crowds. Apart from accurate prediction, efficiency is also an important factor in realistic applications. To overcome the above limitation, we propose a compact model called Spatial-Temporal Consistency Network (STC-Net). In STC-Net, dilated temporal convolutions are introduced to model long-range dependencies along each trajectory for better temporal modeling while graph convolutions are employed to model the spatial interaction among different trajectories. Furthermore, we propose a feature-wise convolution to generate the predicted trajectories in one pass and refine the forecast trajectories together with the reconstructed observed trajectories. We demonstrate that STC-Net generates spatially and temporally consistent trajectories and outperforms other graph-based methods.

1.3.6 Efficient Transformer architecture for motion forecasting



Actually, the high-definition map (HD map) plays an important role in trajectory forecasting. Compared to normal maps, HD maps target autonomous driving and only include necessary task-related information. In this work, we focus on utilizing map information to help trajectory prediction tasks under autonomous driving scenarios. However, modeling all the interactions between different road users and drivable lanes in an efficient way remains a challenging problem. To address this issue, we propose an efficient architecture that introduces an additional aggregation token per lane and trajectory. It allows to modeling of global dependencies within each lane or trajectory and global dependencies between the lanes or trajectories. We thus term the network Aggregation-Interaction Transformer. As a second contribution, our approach learns intention seeds that are used as queries to generate diverse future trajectories in a very efficient way. We conduct comprehensive

experiments and our method achieves state-of-the-art performance both in terms of accuracy and efficiency.

1.4 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2 provides an overview of the related work on LiDAR point cloud semantic segmentation and motion forecasting. It also provides a brief summary of the datasets that are used to evaluate the proposed approaches.

Chapter 3 gives a brief overview of the concepts that are used throughout the thesis. It starts with a quick introduction to neural networks, including convolutional and recurrent networks. Then, it describes the transformer architecture. Finally, it provides a formal definition of the evaluation metrics that are used to evaluate the performance of the proposed approaches.

Chapter 4 proposes a projection-based method, called Multi-scale Interaction Network (MINet), which is very efficient and accurate. This chapter is based on (*Li et al.*, 2021a).

Chapter 5 extends the approach proposed in Chapter 3 to predict the moving status of each point in the LiDAR point cloud. Additionally, we created a new benchmark for LiDAR-based moving object segmentation based on SemanticKITTI. This chapter is based on (*Chen et al.*, 2021b).

Chapter 6 introduces the general concept of reformulating 3D point-based operations such that the point-based methods can operate in the projection space. This chapter is based on (*Li et al.*, 2021b).

Chapter 7 incorporates a temporal fusion layer to extract useful information from previous scans and integrate it with the current scan to solve the "many-to-one" problem caused by the range image's limited horizontal and vertical angular resolution. It further proposes a max-voting-based post-processing technique to correct false predictions.

Chapter 8 proposes an efficient architecture that enforces temporal consistency between historical data and predicted future data for human trajectory prediction. This chapter is based on (*Li et al.*, 2021c).

Chapter 9 proposes an efficient architecture that introduces an additional aggregation token per lane and trajectory for motion forecasting in autonomous driving.

Finally, conclusions are given in Chapter 10 along with suggested directions for future work.

CHAPTER 2

Related Work

In this chapter, we discuss the related literature to this thesis. As this thesis mainly includes two subtopics, LiDAR point cloud semantic segmentation and trajectory forecasting, we split this chapter into two parts and describe each part in detail.

In the first part, we discuss recent progress in LiDAR point cloud semantic segmentation. The LiDAR point cloud has different data representations, like the original 3D point cloud, 2D projection map, which projects the 3D LiDAR point cloud into 2D space by spherical projection, and 3D voxels, which voxelize the 3D point cloud into 3D voxels. To handle different data representations, different methods are proposed. Thus we categorize these methods according to the data representation they adopted and name them point-based methods, projection-based methods, and voxel-based methods respectively. We also discuss some fusion-based methods that try to utilize the benefits of different representations. Although most works target single LiDAR scan processing, more and more works focus on utilizing historical data. Unfortunately, most of these works are for moving object segmentation. We also include a discussion of these methods. Finally, post-processing is an important step for projection-based methods, we will briefly discuss its recent progress.

In the second part, we introduce the recent progress in trajectory prediction. We demonstrate the motion forecasting methods with or without the usage of map information. For the low-speed scenario, like human trajectory prediction, most methods do not utilize the map but can still achieve good performances. By incorporating the map information, the performance will be improved to some extent. However, autonomous driving cars usually lie in a highly dynamic environment where most cars are at high speed. In this case, the map can play a more important role. This is because the drivers usually follow some driving rules, like the cars usually follow the lanes, thus the map can provide meaningful guidance information. In this chapter, we discuss the literature for both mapless and map-assisted motion forecasting methods.

Contents

2.1 LiDAR point cloud semantic segmentation

2.1.1 Point-based methods

As the original data representation of the LiDAR point cloud is 3D point cloud, most methods for processing 3D point cloud can be directly applied here. PointNet (*Qi et al.*, 2017b) is the first work that introduces deep learning into point cloud processing. It directly consumes point clouds and respects the permutation invariance of points in the input. It provides a unified architecture for applications ranging from object classification, and part segmentation, to scene semantic parsing. As its following work, PointNet++ (*Qi et al.*, 2017d) is a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distances, the network learns local features with increasing contextual scales. KPConv (*Thomas et al.*, 2019c) is a new design of point convolution that operates on point clouds without any intermediate representation. The convolution weights of KPConv are located in Euclidean space by kernel points and applied to the input points close to them. KPConv can also be extended to deformable convolutions that learn to adapt kernel points to local geometry. Tangent Convolutions (*Tatarchenko et al.*, 2018) introduces tangent convolutions, a new method for convolutional networks on 3D data. It operates directly on surface geometry and is applicable to unstructured point clouds.

Instead of directly processing the point cloud in 3D space, some methods map 3D coordinates into a high dimension of space. LatticeNet (*Rosu et al.*, 2019) utilizes a sparse permutohedral lattice. It uses PointNet to describe the local geometry of point clouds and embeds them into the lattice for fast convolutions. It also introduces DeformSlice, a learned data-dependent interpolation for projecting lattice features back onto the point cloud. SPLATNet (*Su et al.*, 2018) directly operates on a collection of points represented as a sparse set of samples in a high-dimensional lattice. It uses sparse bilateral convolutional layers as building blocks to maintain efficiency and allow flexible specifications of the lattice structure enabling hierarchical and spatially-aware feature learning.

However, the above methods are not designed for LiDAR point cloud which is usually very sparse while with massive points. Recently some point-based methods target LiDAR point cloud semantic segmentation. SPGs (*Landrieu and Simonovsky*, 2018) presents a novel deep learning-based framework for semantic segmentation of large-scale point clouds of millions of points using superpoint graphs (SPGs). SPGs offer a compact yet rich representation of contextual relationships between object parts, which is then exploited by a graph convolutional network. *Hu et al.* (2020a) present a simple and efficient neural architecture for semantic segmentation of large-scale 3D point clouds. It uses random point sampling instead of more complex point selection approaches and introduces a novel local feature aggregation module to increase the receptive field for each 3D point.

2.1.2 Voxel-based methods

Although the above point-based methods can produce reasonable outcomes, the performance is far from satisfactory. This is because the LiDAR point cloud has its own characteristics, like large-scale and sparse. Unfortunately, most point-based methods are not optimized for these characteristics and thus cannot perform well. Because the LiDAR point cloud is usually very sparse. In other words, most space is empty. SPVCNN (*Tang et al.*, 2020) observes this and introduces Sparse Point-Voxel Convolution (SPVConv), a lightweight 3D module that equips the vanilla Sparse Convolution with a high-resolution point-based branch. This point-based branch preserves fine details even from large

outdoor scenes. It further presents 3D Neural Architecture Search (3D-NAS) to search the optimal network architecture over a diverse design space efficiently and effectively. To produce a dense prediction of the whole scene, JS3C-Net (*Yan et al.*, 2021) tries to fuse semantic segmentation and semantic scene completion (SSC) together. By merging multiple frames in the LiDAR sequence as supervision, the optimized SSC module has learned the contextual shape priors from sequential LiDAR data, completing the sparse single sweep point cloud to the dense one.

However, splitting LiDAR point clouds as cubics does not follow the data capture characteristic of the LiDAR sensor. To solve this issue, PolarNet (*Zhang et al.*, 2020b) proposes an Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. It presents a new LiDAR-specific, nearest-neighbor-free segmentation algorithm. Instead of using common spherical or bird's-eye-view projection, PolarNet's polar bird's-eye-view representation balances the points across grid cells in a polar coordinate system. Cylinder (*Zhu et al.*, 2021) proposes a new framework for outdoor LiDAR segmentation, where cylindrical partition and asymmetrical 3D convolution networks are designed to explore the 3D geometric pattern while maintaining inherent properties of outdoor point clouds. The proposed model acts as a backbone and the learned features from this model can be used for downstream tasks such as point cloud semantic and panoptic segmentation or 3D detection. (AF)2-S3Net (*Cheng et al.*, 2021b) proposes an end-to-end encoder-decoder CNN network for 3D LiDAR semantic segmentation. It introduces a novel multi-branch attentive feature fusion module in the encoder and a unique adaptive feature selection module with feature map re-weighting in the decoder. This method aims to address issues such as high computational complexity and lack of fine details of smaller instances present in other methods. Spherical Transformer (*Lai et al.*, 2023) introduces a spherical partition and a spherical transformer module that can capture the global and local geometric patterns of point clouds.

2.1.3 Projection-based methods

Efficiency is of vital importance for autonomous driving. Although applying sparse convolution on 3D voxels can reduce the computation a lot, it is still not good enough for realistic applications as the operations are still conducted in 3D space. With spherical projection introduced in the following chapter, the LiDAR point cloud can be converted into a 2D projection map. Thus one solution for LiDAR point cloud semantic segmentation is conducting semantic segmentation on a 2D projection map and then unproject predictions back into 3D space. RangeNet++ (*Milioto et al.*, 2019b) exploits range images as an intermediate representation in combination with a Convolutional Neural Network (CNN) that takes into account the rotating LiDAR sensor model. To obtain accurate results, a novel post-processing algorithm is introduced that addresses issues such as discretization errors and blurry CNN outputs. The method can perform full semantic segmentation of LiDAR point clouds at the sensor frame rate. SalsaNext (*Cortinhal et al.*, 2020c) presents an encoder-decoder network for the uncertainty-aware semantic segmentation of a full 3D LiDAR point cloud in real time. SalsaNext is an improved version of SalsaNet, which introduces a new context module, a new residual dilated convolution stack, a pixel-shuffle layer, and a Lovasz-Softmax loss. SalsaNext also injects a Bayesian treatment to compute the epistemic and aleatoric uncertainties for each point in the cloud. SqueezeSeg (*Wu et al.*, 2018b) is an end-to-end pipeline based on convolutional neural networks (CNN) that takes a transformed LiDAR point cloud as input and directly outputs a point-wise label map, which is then refined by a conditional random field (CRF) implemented as a recurrent

layer. SqueezeSegV2 (*Wu et al.*, 2019b) improve the robustness of SqueezeSeg (*Wu et al.*, 2018b). With improved model structure, training loss, batch normalization, and additional input channels, SqueezeSegV2 achieves significant accuracy improvement when trained on real data. It also proposes a domain-adaptation training pipeline for better performance. SqueezeSegV3 (*Xu et al.*, 2020b) proposes Spatially-Adaptive Convolution (SAC) to adopt different filters for different locations according to the input image. SAC can be computed efficiently and is a general framework such that several previous methods can be seen as special cases of SAC. 3D-MiniNet (*Alonso et al.*, 2020) is a fast and efficient method for semantic segmentation of LIDAR point clouds. It first learns a 2D representation from the raw points through a novel projection that extracts local and global information from the 3D data. This representation is fed to an efficient 2D Fully Convolutional Neural Network (FCNN) that produces a 2D semantic segmentation. FPS-Net (*Xiao et al.*, 2021) exploits the uniqueness and discrepancy among the projected image channels for optimal point cloud segmentation. FPS-Net adopts an encoder-decoder structure and uses a residual dense block with multiple receptive fields as a building block in the encoder which preserves detailed information in each modality and learns hierarchical modality-specific and fused features effectively. KPRNet (*Kochanov et al.*, 2020a) adopts recent advances in both image and point cloud segmentation to achieve better accuracy in segmenting LiDAR scans. KPRNet improves the convolutional neural network architecture of 2D projection methods and utilizes KPConv to replace commonly used post-processing techniques with a learnable point-wise component which allows for obtaining more accurate 3D labels. Most above methods adopt the convolution neural network (CNN) architecture. Rangeformer (*Kong et al.*, 2023a), instead, introduces the transformer architecture (*Vaswani et al.*, 2017b) for this task. Besides, it also investigates the challenges and limitations of using range view projections for LiDAR semantic and panoptic segmentation. It identifies three key factors that affect the performance of range view models: many-to-one mapping, semantic incoherence, and shape deformation.

2.1.4 Fusion-based methods

Recently, there are some methods that fuse information from different views or modalities to further improve performance. 2DPASS (*Yan et al.*, 2022) aims to improve the representation learning of point clouds by leveraging rich appearance information from 2D images. The paper introduces a general training scheme that uses an auxiliary modal fusion and multi-scale fusion-to-single knowledge distillation (MSFSKD) to transfer the knowledge from multi-modal data to the pure 3D network. Point-to-Voxel Knowledge Distillation (*Hou et al.*, 2022) addresses the problem of distilling knowledge from a large teacher model to a slim student network for LiDAR semantic segmentation. It proposes Point-to-Voxel Knowledge Distillation (PVD), which transfers the hidden knowledge from both the point level and voxel level. It also introduces a difficulty-aware sampling strategy and an inter-point and inter-voxel affinity distillation to better exploit the structural information of point clouds. MPF (*Alnaggar et al.*, 2021) introduces a novel approach for 3D point cloud semantic segmentation that exploits multiple projections of the point cloud to mitigate the loss of information inherent in single projection methods. It uses two separate highly efficient 2D fully convolutional models to analyze spherical and bird's-eye view projections and then combines the segmentation results of both views. GFNet (*Qiu et al.*, 2022) explores the geometric correspondence between different views in an align-before-fuse manner. A novel geometric flow module (GFM) is devised to bidirectionally align and propagate the complementary information across different views according

to geometric relationships under the end-to-end learning scheme.

Multi-frame information plays a crucial role in LiDAR data processing. For example, MOS (*Chen et al.*, 2021c) and MotionSeg3D (*Sun et al.*, 2022a) generate residual images from multiple LiDAR frames to explore the sequential information and use it for segmenting moving and static objects. Motivated by these approaches, Meta-RangeSeg (*Wang et al.*, 2022b) also uses residual range images for the task of semantic segmentation of LiDAR sequences. It employs a meta-kernel to extract the meta-features from the residual images. SeqOT (*Ma et al.*, 2022a) exploits sequential LiDAR frames using yaw-rotation-invariant OverlapNets (*Chen et al.*, 2020, 2021a) and transformer networks (*Vaswani et al.*, 2017a; *Ma et al.*, 2022b) to generate a global descriptor for fast place recognition in an end-to-end manner. In addition, SCPNet (*Xia et al.*, 2023) designs a knowledge distillation strategy between multi-frame LiDAR scans and a single-frame LiDAR scan for semantic scene completion. Recently, Mars3D (*Liu et al.*, 2023a) designed a plug-and-play motion-aware module for multi-scan 3D point clouds to classify semantic categories and motion states. Seal (*Liu et al.*, 2023b) proposes a temporal consistency loss to constraint the semantic prediction of superpoints from multiple scans. Although the benefit of using multiple scans has been studied, these works address other tasks.

Although projection-based LiDAR segmentation methods are computationally efficient, they suffer from boundary blurriness or the many-to-one issue (*Wu et al.*, 2018a; *Milioto et al.*, 2019c). To alleviate this issue, most works use a conditional random field (CRF) (*Wu et al.*, 2018a) or k-NN (*Milioto et al.*, 2019c) to smooth the predicted labels. *Wu et al.* (2018a) implements the CRF as an end-to-end trainable recurrent neural network to refine the predictions according to the predictions of the neighbors within three iterations. It does not address occluded points explicitly. k-NN (*Milioto et al.*, 2019c) infers the semantics of ambiguous points by jointly considering its k closest neighbors in terms of the absolute range distance. However, finding a balance between under and over-smoothing can be challenging, and it may not be able to handle severe occlusions. Recently, NLA (*Zhao et al.*, 2021) assigns the nearest point's prediction in a local patch to the occluded point. However, it is required to iterate over each point to verify occlusions. In addition, RangeFormer (*Kong et al.*, 2023b) addresses this issue by creating sub-clouds from the entire point cloud and inferring labels for each subset. However, partitioning the cloud into sub-clouds ignores the global information. It can also not easily be applied to existing networks. Some methods (*Kochanov et al.*, 2020b; *Sun et al.*, 2022a; *Ando et al.*, 2023) propose additional refinement modules for the networks to refine the initial estimate, which increases the runtime. In this work, we propose to tackle this issue by combining past predictions in an efficient max-voting manner. Our method complements existing approaches and can be applied to various networks.

2.2 Motion Forecasting

2.2.1 Trajectory Forecasting without Maps

One branch of trajectory forecasting approaches focuses on agent interactions while not utilizing nearby map information (*Xu et al.*, 2022; *Yue et al.*, 2022; *Huang et al.*, 2019; *Alahi et al.*, 2016; *Mohamed et al.*, 2020). These methods are usually applied in low-speed scenarios with pedestrians. RNN-based methods model the motion of each pedestrian by a recurrent architecture and the context of nearby trajectories by pooling operations. Social-LSTM (*Alahi et al.*, 2016) is one of the earli-

est methods following this approach and several methods explore different strategies to improve the performance. PIF (*Liang et al.*, 2019) takes human behavioral information into consideration and introduces rich visual features. SR-LSTM (*Zhang et al.*, 2019b) proposes a social-aware information selection mechanism to select useful information from neighboring pedestrians. FvTraj (*Bi et al.*, 2020) focuses on first-person view-based trajectory forecasting with multi-head attention mechanisms. Recently, *Mangalam et al.* (2020) proposed a variational auto-encoder to predict first the endpoints of the trajectories and then the corresponding trajectories in a second step. While it achieves impressive results, it is too expensive for most applications.

GAN-based methods take into account that many future trajectories might be plausible. To handle this problem, Social-LSTM (*Alahi et al.*, 2016) is converted into a generative model in Social-GAN (*Gupta et al.*, 2018). Based on it, SoPhie (*Sadeghian et al.*, 2019) combines a social attention mechanism with a physical attention mechanism. CGNS (*Li et al.*, 2019) replaces the LSTMs in SoPhie (*Sadeghian et al.*, 2019) by GRUs. Similar to (*Mangalam et al.*, 2020), Goal-GAN (*Dendorfer et al.*, 2020) uses a two-stage process for trajectory forecasting. Since these works also rely on recurrent architectures, they are as inefficient as RNN-based methods. Graph-based methods use graph convolutions instead of simple pooling operations for modeling spatial interactions. Social-BiGAT (*Kosaraju et al.*, 2019) is a graph-based generative model that uses a graph attention network to encode social interactions between pedestrians and a recurrent encoder-decoder architecture to predict future trajectories. Similar to Social-BiGAT (*Kosaraju et al.*, 2019), STGAT (*Huang et al.*, 2019) captures spatial interactions by a graph attention mechanism at each time step and uses LSTMs for forecasting. RSBG (*Sun et al.*, 2020) proposes a social behavior graph that encodes the social representations. DAG-Net (*Monti et al.*, 2020) uses a double attentive graph neural network for trajectory forecasting. These methods, however, remain inefficient since they still use a recurrent architecture. Social-STGCNN (*Mohamed et al.*, 2020) omits the RNNs and uses a CNN for time extrapolation. While this results in a very low latency, it comes at the cost of a very weak temporal model.

Recent advancements in trajectory prediction have seen the application of encoder-decoder structures, as illustrated in studies by *Cheng et al.* (2021a); *Mangalam et al.* (2020); *Sun et al.* (2021). These structures are favored for their ability to effectively encode a range of contextual features. The MID framework, introduced by *Gu et al.* (2022), presents a novel stochastic approach incorporating motion indeterminacy diffusion. This method redefines trajectory prediction as evolving from a general walkable area to a specific desired trajectory. In a different approach, Memonet developed by *Xu et al.* (2022), departs from traditional parameter-based frameworks that rely on optimizing model parameters through training data. Instead, Memonet employs an instance-based framework, utilizing retrospective memory to store and reference various past trajectories and their associated intentions.

2.2.2 Trajectory Forecasting with Maps

For autonomous driving, the map information plays an important role as it can provide a strong prior for the future trajectory (*Mangalam et al.*, 2021; *Wang et al.*, 2022a; *Mangalam et al.*, 2020). The map information can be represented as an image, which is a semantic bird-view map, or as a lane map, which represents each lane by a sequence of points. Early research in this field applied long short-term memory (LSTM) networks (*Alahi et al.*, 2016) to model historical states of agents, and convolutional neural networks (CNNs) for processing rasterized scene images (*Cui et al.*, 2019;

Gilles et al., 2021a; *Huang et al.*, 2022b; *Salzmann et al.*, 2020). To capture agent interactions, graph neural networks (GNNs) have been extensively used, as seen in works by *Chen et al.* (2022); *Gilles et al.* (2022); *Huang et al.* (2022a); *Mo et al.* (2022) , which represent these interactions through scene or interaction graphs. More recently, the Transformer encoder-decoder architecture has emerged as a popular method for motion prediction, exemplified by the SceneTransformer (*Ngiam et al.*, 2021) and WayFormer (*Nayakanti et al.*, 2023), known for their concise model structure and enhanced performance. However, existing Transformer-based models have primarily focused on the encoding process, with less attention to decoding. Addressing this, the Motion Transformer (*Shi et al.*, 2022) proposed an advanced decoding stage that utilizes iterative local motion refinement for improved prediction accuracy. In the realm of using prediction models for planning tasks, there's a significant focus on multi-agent joint motion prediction frameworks, as discussed in studies by *Gilles et al.* (2021b); *Jia et al.* (2022); *Mo et al.* (2022); *Sun et al.* (2022b). These frameworks are crucial for efficient and consistent prediction of diverse, multi-agent trajectories. A common challenge with current motion prediction models is their tendency to overlook the impact of autonomous vehicle (AV) actions, which limits their applicability in subsequent planning tasks. To address this, researchers have developed conditional multi-agent motion prediction models that incorporate AV planning data into the prediction mechanism, as seen in works by *Espinoza et al.* (2022); *Huang et al.* (2023b); *Song et al.* (2020).

2.2.3 Forecasting other Modalities

In the context of autonomous driving, there are also other approaches that do not forecast trajectories, but other modalities (*Hu et al.*, 2021a; *Akan and Güney*, 2022; *Zhang et al.*, 2022) combine perception, sensor fusion, and prediction and forecast future instance segmentation maps in the bird eye view (BEV). Apart from object-level forecasting, some methods focus on scene-level prediction, like occupancy forecasting or semantic scene completion. Approaches for occupancy forecasting (*Hu et al.*, 2022b, 2021b; *Khurana et al.*, 2022; *Shi et al.*, 2023) use a free space-centric representation, which can be directly obtained by raycasting. This representation can capture the free space of the surrounding scene and can assist in forecasting tasks. Some methods not only estimate the binary occupancy map but aim to reconstruct the entire semantic scene (*Behley et al.*, 2019b; *Garbade et al.*, 2019; *Huang et al.*, 2023a; *Cao and de Charette*, 2022), which is also named semantic scene completion. There are also some works that focus directly on end-to-end driving (*Prakash et al.*, 2021; *Chitta et al.*, 2021; *Hu et al.*, 2022a).

CHAPTER 3

Preliminaries

In this chapter, we first delve into fundamental concepts pertinent to this thesis. We commence with an overview of foundational techniques employed, encompassing Neural Networks, Convolutional Neural Networks, and Transformer architecture. Furthermore, we outline the two primary tasks addressed in this thesis: LiDAR Point Cloud Semantic Segmentation and Motion Forecasting. The used datasets and the metrics for evaluation are also presented in the end.

Contents

3.1 Neural Networks

A neural network is a computational model structured around interconnected layers, with each successive layer receiving inputs from its predecessor. These layers house multiple processing units known as neurons. These neurons perform transformations on the data relayed from the previous layer. Broadly, the layers of a neural network can be classified into three categories: the input layer, hidden layers, and the output layer. The network begins with the input layer and culminates in the

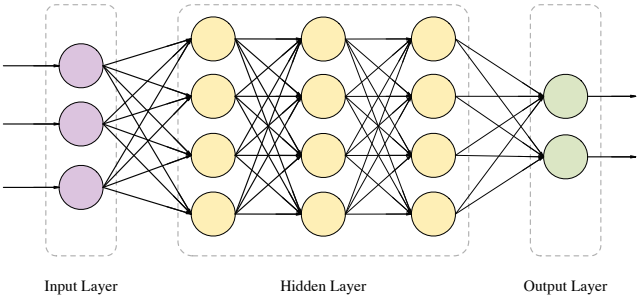


Figure 3.1: A feed-forward neural network comprises two hidden layers. Within each layer are multiple neurons, and every neuron establishes connections with all neurons from the preceding layer.

output layer, with one or more hidden layers sandwiched between them. In its canonical form, the nodes and connections in neural networks create a directed acyclic graph. Such configurations are termed feed-forward neural networks or multi-layer perceptrons (MLP). Figure 3.1 illustrates the architecture of a feed-forward neural network featuring two hidden layers. The primary utility of neural networks is function approximation, particularly in pattern recognition domains where defining explicit rule-based solutions is impractical. Subsequently, we will provide a concise overview of the integral components of neural networks.

3.1.1 Neuron

The neuron stands as the foundational element of neural networks. Conceptually, it operates as a system that ingests inputs undergoes a transformation process, and subsequently emits an output, termed the "activation". Initially, it calculates a weighted sum of its inputs, integrates a bias, and then channels this aggregate through an activation function to yield an output. This activation function is typically a differentiable, non-linear function. Feed-forward neural networks are architecturally assembled by layering these neurons. In such a configuration, each neuron determines a weighted sum of the activations from its preceding layer. This sum then undergoes a non-linear transformation, producing an activation that serves as input for the subsequent layer. The process is described as follows:

$$o_j^{(l)} = b_j^{(l)} + \sum_{i=1}^{K} w_{ij}^{(l)}\, a_i^{(l-1)}, \tag{3.1}$$

$$a_j^{(l)} = f(o_j^{(l)}), \tag{3.2}$$

where $o_j^{(l)}$ is the weighted sum of activations from layer $l-1$, $a_j^{(l)}$ is the activation of neuron j at layer l, $w_{ij}^{(l)}$ is the weight from neuron i at layer $l-1$ to neuron j on layer l, K is the number of neurons at layer $l-1$. In Equation (3.2), f is a non-linear activation function.

The incorporation of non-linearity in intermediate layers is imperative. Absent this non-linear feature, the entire network's cumulative transformation capability would simply mirror that of a singular layer. Figure 3.3 delineates the structure of a neuron, highlighting its inputs, transformations,

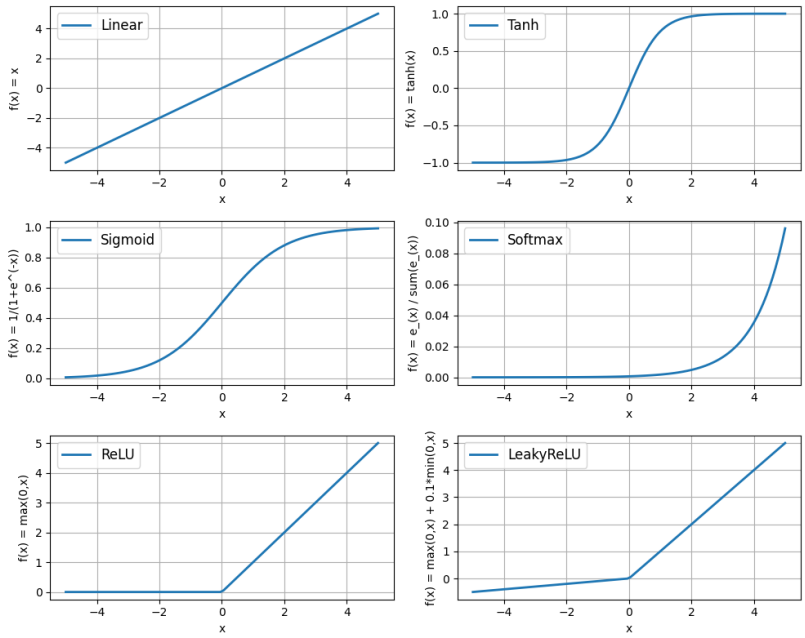


Figure 3.2: Different activation functions: Linear, Rectified Linear Unit (ReLU), tanh, and sigmoid.

and output. Conversely, Figure 3.2 showcases a selection of prevalent activation functions frequently utilized in neural networks.

3.1.2 Output Layer

The output layer represents the terminal layer of the neural network, producing the final prediction corresponding to a given input. The choice of activation function for this layer hinges on the specific nature of the task at hand. For regression tasks, where the desired output is a continuous real number, a linear activation function is typically employed.

$$f_{linear}(x) = x, \tag{3.3}$$

For classification tasks, the choice of activation function is influenced by the number of classes. Specifically, in binary classification scenarios where there are only two classes, a sigmoid activation function is utilized.

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{3.4}$$

The neuron's output is scaled to fall between 0 and 1, representing the probability of the corresponding class, in scenarios such as binary classification. Likewise, for multi-class classification tasks, a softmax layer is employed to ensure that the network's output forms a valid probability distribution across the various classes.

$$f_{softmax}(o_i) = \frac{e^{o_i}}{\sum_j e^{o_j}}, \tag{3.5}$$

where o_i is the output of the $i\text{-}th$ neuron in the output layer. It's important to highlight that the softmax function is employed exclusively for single-label classification tasks. In contrast, for multi-label classifications, sigmoid activation is the preferred choice.

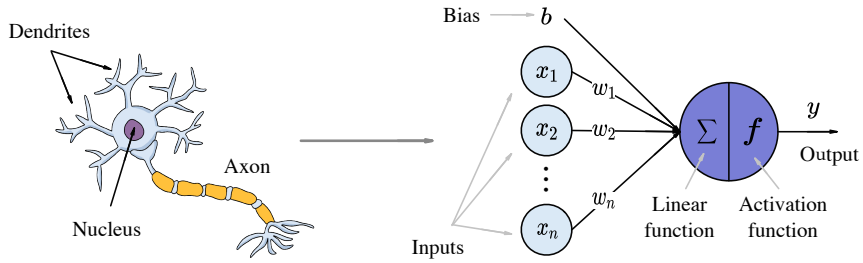


Figure 3.3: A neuron with n inputs $(x_1, \ldots, x_n)$, weights $(w_1, \ldots, w_n)$, a bias (b) and activation function f.

Additionally, the activation function of the output layer can be strategically chosen to impose specific constraints on the output values. For example, an exponential activation ensures predictions are positive, while a tanh activation confines the network's output to a range between -1 and 1.

3.1.3 Objective Function

The objective function quantifies the divergence between the neural network's predictions and the expected results. Often referred to as the loss or error function, it serves as a barometer for model performance. This function steers the neural network training, directing it towards an optimal set of parameters that diminish the error. This is achieved by progressing in the direction opposite to the gradients of the objective function. Hence, it's imperative for the objective function to be differentiable concerning the network's outputs. A prevalent choice for an objective function is the mean squared error (MSE).

$$\mathcal{L}_{mse}(\theta) = \frac{1}{N}\sum_{n=1}^{N}(y(x_n, \theta) - t_n)^2, \tag{3.6}$$

the objective function $\mathcal{L}_{mse}$ is characterized by the network parameters θ (comprising the weights and biases of the network). Here, N denotes the total number of training examples, $y(x_n, \theta)$ represents the output corresponding to the $n-th$ training example, and t_n is the target or desired value for that particular example. For multi-class classification tasks encompassing C classes, the cross-entropy loss is typically utilized.

$$\mathcal{L}_{ce}(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} t_{n,c}\, log(y_c(x_n, \theta)), \tag{3.7}$$

the target t_n is depicted through a one-hot encoding of the target class, with $t_{n,c}$ being its value for class c. Likewise, $y_c(x_n, \theta)$ denotes the predicted probability for class c. Given that the target is represented as a one-hot encoding vector, this loss can be succinctly expressed as:

$$\mathcal{L}_{ce}(\theta) = -\frac{1}{N}\sum_{n=1}^{N} log(y_c(x_n, \theta)), \tag{3.8}$$

where $y_c(x_n, \theta)$ is the predicted probability for the target class c.

Broadly speaking, any function that gauges the deviation between the network's output and a designated target value can serve as an objective function for training a neural network, provided it's differentiable in relation to the network's parameters.

3.1.4 Network Training

In earlier discussions, we delved into how neural networks employ layers of transformations to correlate inputs with the anticipated outputs. The essence of these transformations, their parameters, are deduced from data by scouring the parameter space to find the optimal set of parameters, encompassing both weights and biases, that curtail the objective function.

Typically, gradient descent-based methodologies are employed to train neural networks. Beginning with a preliminary parameter set, the training algorithm iteratively refines these parameters, nudging them in the direction contrary to the gradients of the objective function relative to the network's parameters

$$\theta_{i+1} = \theta_i - \eta \nabla \mathcal{L}(\theta_i) \tag{3.9}$$

In this formulation, θ_i denotes the network parameters at the $i\text{-}th$ iteration, while $\mathcal{L}$ represents the objective function. The symbol η stands for the learning rate, a small positive number that dictates the extent of movement in the direction of the negative gradients. To determine the gradients relative to the parameters in the intermediate layers, the backpropagation algorithm is employed. This algorithm recursively utilizes the chain rule, ensuring that the gradients concerning the parameters at layer l are derived based on the gradients at layer $l + 1$

The selection of the learning rate, denoted as η, is pivotal for successful training convergence. Setting η too high can lead to oscillatory behavior around the optimal value. Conversely, an overly conservative η can protract the convergence process, making training inordinately lengthy. Contemporary methods employ adaptive learning rates tailored for individual parameters, drawing upon gradient moments; an exemplar of this is the Adam optimizer, as highlighted in (*Kingma and Ba*, 2015).

3.2 Convolutional Neural Networks (CNNs)

Conventional neural networks predominantly feature fully connected layers, where every neuron is linked to all the neurons from the preceding layer. Consequently, each layer is essentially a matrix multiplication of a weight matrix and the activation vector of the prior layer. In contrast, convolutional networks incorporate at least one convolutional layer. Unlike their fully connected counterparts, convolutional layers manifest sparse connectivity, with each neuron tethered only to a localized region from the preceding layer. Additionally, neurons within the same convolutional layer share parameters, leading to a substantial decrease in the total parameter count when compared to fully connected layers.

Convolutional neural networks (CNNs) have gained significant traction in image processing, demonstrating remarkable efficacy in image classification and recognition tasks. Each layer within a CNN can be conceptualized as a learned filter or kernel that distills local features from the input image. Owing to the shared parameters across neurons focusing on different regions, multiple kernels can be learned with a relatively modest parameter set. At its core, a convolutional layer performs a

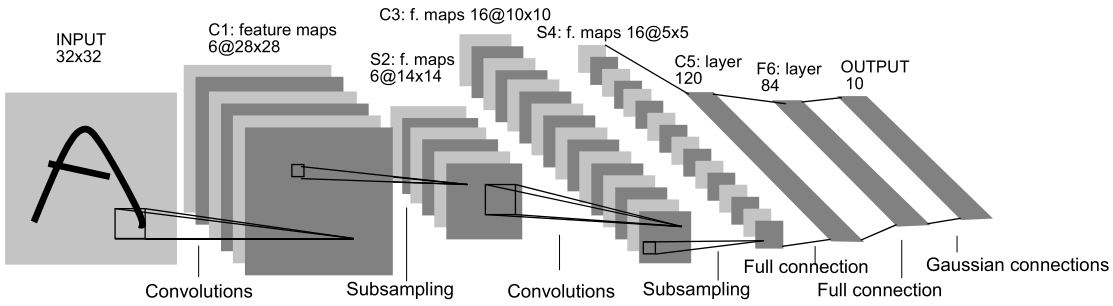


Figure 3.4: The network architecture of LeNet-5. (*LeCun et al.*, 1998).

convolution operation between the input image I and the learned kernel W. This two-dimensional convolution is defined as:

$$(I * W)(i, j) = \sum_{l,m} I(i - l, j - m)W(l, m). \tag{3.10}$$

Typically, convolutional networks comprise a sequence of convolutional layers, succeeded by fully connected layers towards the end. Following the convolutional stages, a non-linear transformation is enacted, often employing activation functions like the ReLU. Occasionally, a pooling layer is incorporated, which consolidates the activations of several neurons into a singular output. Implementing such layers, such as max pooling, bestows the network with resilience to minor translational variations.

Several convolutional neural network architectures have made notable impacts in the domain. Among the seminal 2D CNNs, LeNet-5 (*LeCun et al.*, 1998) stands out, showcased in Figure 3.4. This pioneering convolutional network was instrumental in recognizing handwritten digits. Subsequent architectures like AlexNet (*Krizhevsky et al.*, 2012) and VGG (*Simonyan and Zisserman*, 2015) also gained prominence, proving their mettle in a range of tasks, including image classification and object detection. In contemporary times, architectures drawing inspiration from ResNet (*He et al.*, 2016a) are predominantly favored.

3.3 Transformer

The Transformer architecture follows an encoder-decoder structure, characteristic of numerous Neural Machine Translation models (*Bahdanau et al.*, 2015). Subsequent adaptations of the Transformer, such as the encoder-exclusive BERT or the decoder-centric GPT, have demonstrated exemplary performance in language modeling tasks.

3.3.1 Attention and Self-Attention

Attention mechanisms allow neural networks to enhance predictions by focusing selectively on specific segments of data. This selective focus is quantified through learned weights, resulting in the output typically manifesting as a weighted average.

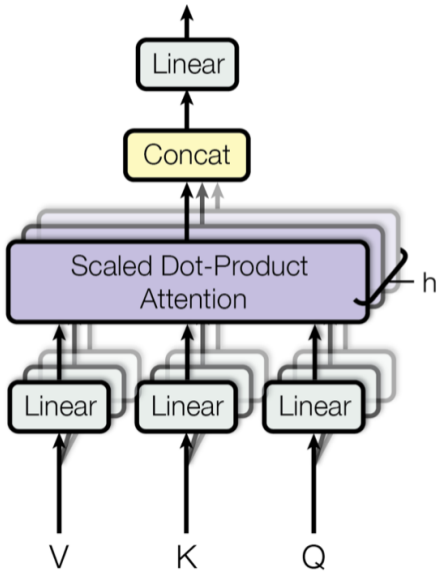


Figure 3.5: Illustration of the multi-head scaled dot-product attention mechanism. This image is from *Vaswani et al.* (2017b).

Self-attention extends this concept, enabling the model to make predictions for a portion of a data sample based on information from other parts of the same sample. Conceptually, it bears resemblance to non-local means. It's important to highlight that self-attention is permutation-invariant, signifying that it operates on sets without being influenced by the order of elements.

Numerous variations of attention and self-attention mechanisms exist. The Transformer model, as introduced by *Vaswani et al.* (2017b), employs the scaled dot-product attention. In this mechanism, given a query matrix $\mathbf{Q}$, a key matrix $\mathbf{K}$, and a value matrix $\mathbf{V}$, the output is derived from a weighted summation of the value vectors. Here, the weighting for each value is governed by the dot-product between the query and its corresponding key:

$$\text{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.11}$$

And for a query and a key vector $\mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^d$ (row vectors in query and key matrices), we have a scalar score:

$$a_{ij} = \text{softmax}\left(\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d_k}}\right) = \frac{\exp\left(\mathbf{q}_i \mathbf{k}_j^\top\right)}{\sqrt{d_k}\sum_{r\in\mathcal{S}_i}\exp\left(\mathbf{q}_i \mathbf{k}_r^\top\right)} \tag{3.12}$$

where $\mathcal{S}_i$ is a collection of key positions for the i-th query to attend to.

3.3.2 Multi-Head Self-Attention

The multi-head self-attention mechanism, as Figure 3.5 stands as a cornerstone of the Transformer architecture. Instead of performing a singular attention computation, this method divides the inputs

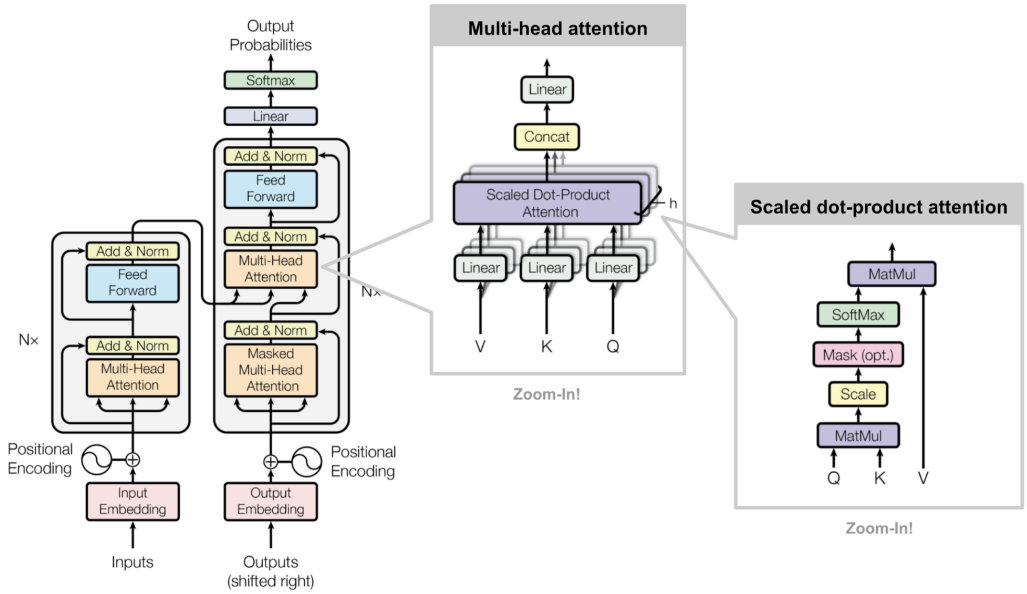


Figure 3.6: The architecture of the vanilla Transformer model. This image is from *Vaswani et al.* (2017b).

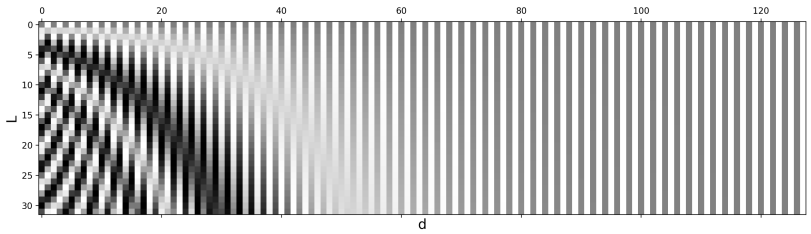


Figure 3.7: Sinusoidal positional encoding with $L = 32$ and $d = 128$. The value is between -1 (black) and 1 (white) and the value 0 is in gray. This image is from *Vaswani et al.* (2017b).

into multiple subspaces and calculates the scaled dot-product attention for each subspace concurrently. The separate attention outcomes are then merged through concatenation and undergo a linear transformation to achieve the desired dimensions.

$$\begin{aligned} \text{MultiHeadAttn}\left(\mathbf{X}_q, \mathbf{X}_k, \mathbf{X}_v\right) &= \left[\text{head}_1; \ldots; \text{head}_h\right] \mathbf{W}^o \\ \text{where head}_{\,i} &= \text{Attention}\left(\mathbf{X}_q \mathbf{W}_i^q, \mathbf{X}_k \mathbf{W}_i^k, \mathbf{X}_v \mathbf{W}_i^v\right) \end{aligned} \tag{3.13}$$

$[.;.]$ denotes a concatenation operation. The matrices $\mathbf{W}_i^q$, $\mathbf{W}_i^k \in \mathbb{R}^{d \times d_k/h}$, and $\mathbf{W}_i^v \in \mathbb{R}^{d \times d_v/h}$ serve as weight matrices to transform input embeddings $\mathbf{X}_q$, $\mathbf{X}_k$, $\mathbf{X}_v$ of dimensions $L \times d$ into query, key, and value matrices, respectively. Additionally, $\mathbf{W}^o \in \mathbb{R}^{d_v \times d}$ represents the output linear transformation. All these weights are to be learned during the training process.

3.3.3 Encoder-Decoder Architecture

The whole transformer architecture is shown in Figure 3.6 which consists of an encoder-decoder architecture.

The encoder produces an attention-oriented representation, pinpointing specific details within a broad context. It's composed of a series of six identical modules. Each of these modules houses two sub-layers: a multi-head self-attention mechanism and a point-wise, fully connected feed-forward network. The term "point-wise" signifies that the same linear transformation, using consistent weights, is applied individually to each element of the sequence. This approach can also be conceptualized as a convolutional layer with a filter size of one. Additionally, every sub-layer incorporates a residual connection alongside layer normalization. Regardless of their functionality, all sub-layers emit data of the consistent dimension d.

In contrast, the Transformer's decoder is tasked with extracting information from the encoded representations. Its structural design bears resemblance to the encoder but introduces a notable difference. Each repeated module in the decoder contains not one, but two multi-head attention sub-layers. The initial multi-head attention sub-layer incorporates a masking mechanism to deter positions from accessing subsequent information, ensuring a logical information flow.

3.3.4 Positional Encoding

Because self-attention operation is permutation invariant, it is important to use proper positional encoding to provide order information to the model. The positional encoding $\mathbf{P} \in \mathbb{R}^{L \times d}$ has the same dimension as the input embedding, so it can be added on the input directly. The vanilla Transformer considered two types of encodings:

3.3.4.1 Sinusoidal Positional Encoding

The sinusoidal positional encoding is articulated as, given the token position $i = 1, \ldots, L$ and the dimension $\delta = 1, \ldots, d$:

$$\mathrm{PE}(i, \delta) = \begin{cases} \sin\left(\frac{i}{10000^{2\delta/d}}\right) & \text{if } \delta \text{ is even} \\ \cos\left(\frac{i}{10000^{28/d}}\right) & \text{if } \delta \text{ is odd} \end{cases} \tag{3.14}$$

Each dimension of the positional encoding corresponds to a distinct sinusoidal wave with varying wavelengths across dimensions, from 2π to $10000\dot{2}\pi$. This is shown in Figure 3.7.

3.3.4.2 Learned Positional Encoding

Learned positional encoding assigns a trainable column vector to each element, signifying its absolute position, as proposed by (*Gehring et al.*, 2017). Additionally, (*Al-Rfou et al.*, 2019) highlighted that this encoding can be adapted individually for each layer.

3.4 LiDAR Point Cloud Semantic Segmentation

3.4.1 LiDAR Data Processing

Over the past two decades, the light detection and ranging (LiDAR) sensor has emerged as a pivotal tool in numerous robotics applications. Inside a LiDAR unit, laser beams discharge pulsed light waves into the environment, then measure the time taken for each pulse to bounce back. This process,

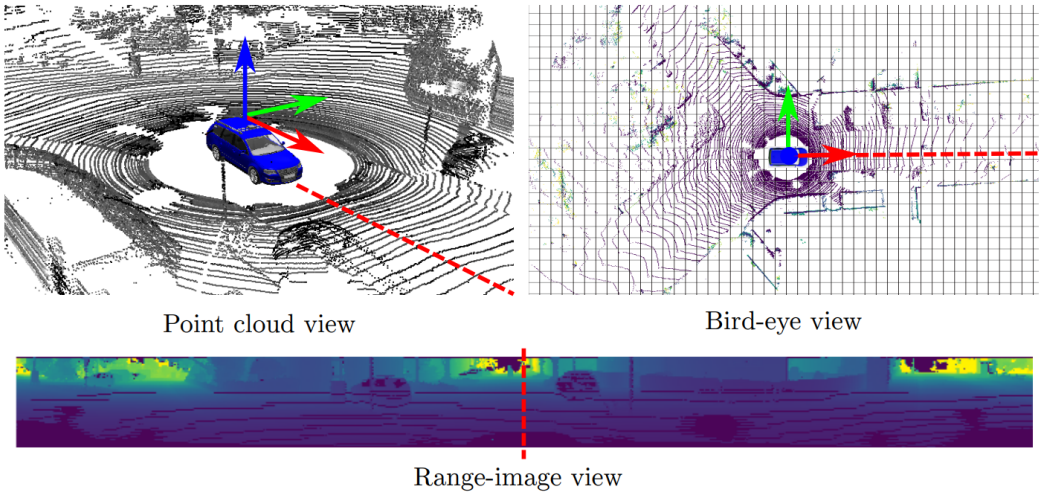


Figure 3.8: Several representations of LiDAR scans.

executed millions of times each second, enables the creation of detailed, real-time 3D environmental maps.

For robotics applications, the market mainly offers two LiDAR sensor variants: mechanically rotating LiDAR and solid-state LiDAR. This thesis predominantly zeroes in on the mechanically rotating variant due to its prevalent use in autonomous vehicles up to the present. Henceforth, references to mechanically rotating LiDAR will simply be termed as "LiDAR". Data collected from a complete rotation of such LiDAR sensors, capturing a full 360-degree environment scan, is denoted as a single LiDAR scan.

Several methods exist to depict a LiDAR scan for subsequent analysis. The three predominant representations include the point cloud view, bird-eye view, and range image view, as showcased in Figure 3.8.

A point cloud, the quintessential output of most LiDAR tools, essentially starts as an amalgamation of range readings paired with sensor orientation metrics. Preliminary processing converts each laser shot's range and orientation into a position within the LiDAR sensor's local 3D coordinate system. This conversion is as follows:

$$
\begin{aligned}
x &= r\cos(\alpha)\sin(\theta), \\
y &= r\cos(\alpha)\cos(\theta), \\
z &= r\sin(\alpha),
\end{aligned}
$$

the triplet $(x, y, z)^\top$ represents the 3D coordinates of a LiDAR point, while r denotes the corresponding range measurement. The symbols θ, α represent the azimuth and inclination orientation angles respectively, as illustrated in Figure 3.11. The resulting point cloud serves as the foundational data for subsequent analysis and processing. Nonetheless, a standard rotating 3D LiDAR sensor can generate several hundred thousand points in a single frame, demanding significant computational resources for processing. The raw point cloud, characterized by its distance-driven sparsity and lack of order, often proves challenging for direct integration into downstream tasks. In particular, methods that leverage neural networks find the task of handling 3D point cloud data daunting. This complexity has yet to be fully addressed, especially for large-scale outdoor applications, including autonomous

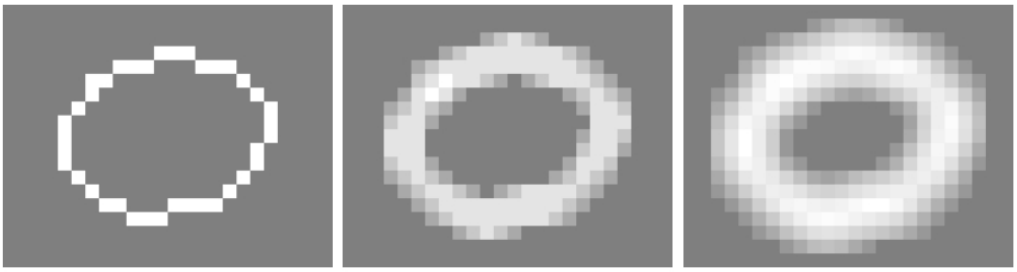


Figure 3.9: Illustration of "submanifold" dilation: On the left is the original curve. The middle shows the outcome of applying a standard 3×3 convolution with weights $1/9$ to it. On the right, the result of applying the same convolution once more is displayed. This example demonstrates that with each convolutional layer, standard convolutions significantly diminish the sparsity of the features. This image is from *Graham et al.* (2018).

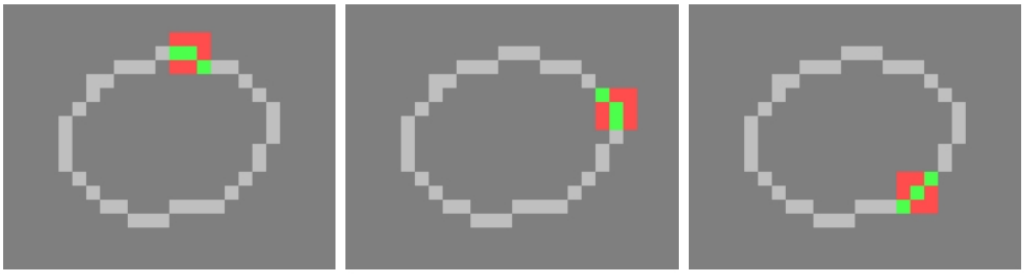


Figure 3.10: The Sparse Submanifold Convolution (SSC) operation, denoted as $\mathrm{SSC}(\cdot, \cdot, 3)$, centers its receptive field at different active spatial locations. Active locations are shown in green, while red locations are ignored by SSC, keeping the pattern of active locations unchanged. This image is from *Graham et al.* (2018).

driving.

The bird-eye view offers a 2D depiction of a LiDAR scan by projecting the point cloud onto a grid map aligned with the x-y plane. Each grid cell captures the height information of its respective point. Given its lightweight nature and its ability to provide a comprehensive top-down perspective, the bird-eye view has become a popular choice for tasks like object detection (*Lang et al.*, 2019), motion planning (*Mohapatra et al.*, 2021), and tracking (*Yin et al.*, 2021). Nevertheless, it has its limitations. When segmenting the space into voxels or pillars, quantization errors can arise, leading to the omission of distant objects represented by only a handful of points. Additionally, this view loses details of vertical structures, such as buildings, trees, and poles, as these 3D objects get flattened onto the 2D x-y plane.

In semantic segmentation, a common alternative to bird-eye view representation is the discretization of LiDAR point clouds into voxels, followed by the application of sparse 3D convolution for processing. This approach is necessitated by the typically large-scale point clouds generated by LiDAR. Processing these voxelized large-scale point clouds directly can be computationally intensive. Furthermore, the volume resulting from voxelizing LiDAR point clouds is often heavily sparse. Utilizing standard 3D convolution on such sparse volumes can lead to a "submanifold" dilation issue, as highlighted by *Graham et al.* (2018) shown in Figure 3.9. This issue can be mitigated by employing

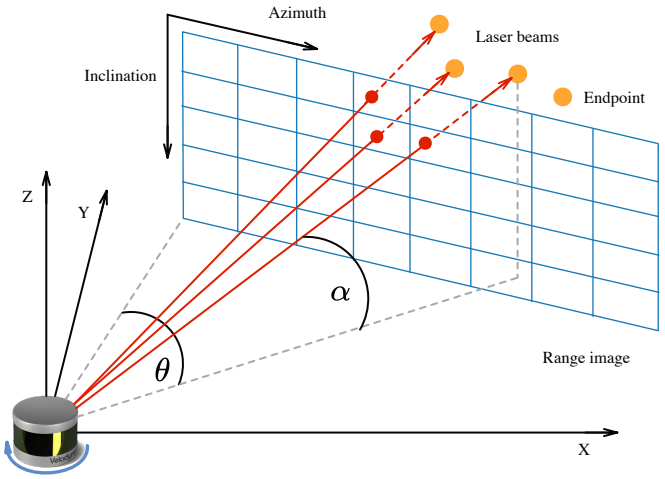


Figure 3.11: Illustrating the generation of a range image from a point cloud. This figure is adapted from the work by *Fan et al.* (2021).

sparse convolution, as proposed by the same authors, which focuses operations only on valid voxels, thereby reducing unnecessary computation shown in Figure 3.10.

The range image view serves as an intuitive representation of scans captured by rotating 3D LiDARs, such as those produced by Velodyne or Ouster sensors. This representation acts as an intermediary step in the transformation of raw LiDAR measurements into a point cloud. As the LiDAR scans, it creates a 2D matrix dictated by its elevation and azimuth resolutions. Each cell within this matrix corresponds to a range measurement, culminating in what's known as the range image R. For a LiDAR configured with h beams and pulsing w times during a single rotation, the resulting range image will have a height of h and a width of w, as depicted in Figure 3.11.

For every pixel in the range image, determined by a specific set of elevation and azimuth angles, the 3D coordinates of the corresponding end-point of the beam can be computed using the below equation. Conversely, starting with a point cloud, a projection transformation can be applied to recreate a range image. Specifically, we convert each LiDAR point $\boldsymbol{p} = (x, y, z)^\top$ via a mapping $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates, and finally to image coordinates, as defined by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right] w \\ \left[1 - \left(\arcsin\left(zr^{-1}\right) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] h \end{pmatrix}$$

the pair $(u, v)^\top$ denotes image coordinates, while h and w represent the height and width of the desired range image representation, respectively. The term $f = f_{up} + f_{\text{down}}$ signifies the vertical field-of-view of the sensor, and $r = \|\boldsymbol{p}_i\|_2$ denotes the range of each point. Following this procedure yields a list of $(u, v)^\top$ tuples, with each tuple containing a pair of image coordinates corresponding to each point $\boldsymbol{p}_i$.

For now, we can categorize LiDAR point semantic segmentation methods into three types according to different data partitions, including point-based methods, projection-based methods and voxel-based methods which are shown in Figure 3.12. In this thesis, we adopt a range projection-based representation for processing LiDAR point cloud data. This approach allows us to leverage the extensive research on 2D convolutional neural networks, which have already proven effective in

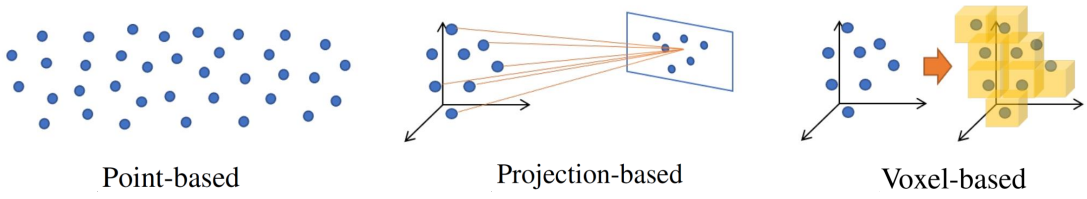


Figure 3.12: Categories of LiDAR semantic segmentation methods. This image is from *Camuffo et al.* (2022).

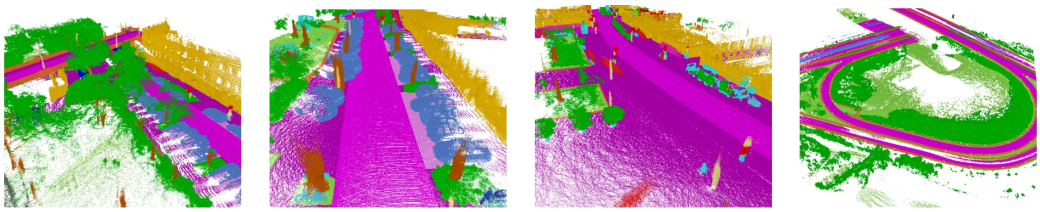


Figure 3.13: Visualization for SemanticKitti dataset. This image is from *Behley et al.* (2019d).

image-related tasks. Processing compact range images is more efficient than handling point clouds directly, and unlike the 2D bird-eye views, there is no loss of raw data information.

An additional advantage of pairing range images with neural networks is the capability to effortlessly incorporate various types of data captured by LiDAR sensors. Using the image indices, for each point $\boldsymbol{p}_i$ we can extract its range r, its coordinates x, y, z and its reflectance value e storing them channel-wise in the image. As a result, each pixel in this representation encapsulates more than just range data. This design facilitates the integration of additional information as extra channels. Thus, we can seamlessly input this augmented data into existing network designs without the need for architectural modifications. This not only enhances the efficacy of our methodologies but also ensures they can be easily adapted to emerging architectures.

3.4.2 Datasets

For LiDAR point cloud semantic segmentation, We use four challenging datasets to evaluate our method:

- **SemanticKITTI** (*Behley et al.*, 2019d) provides pixel-wise semantic labels for the entire KITTI Odometry Benchmark (*Geiger et al.*, 2012b) with more than 20000 scans which are divided into 22 sequences. We use its training set for training (sequences 00-10 without sequence 08), its validation set (sequence 08) for the ablation study, and its test set (sequences 10-21) for comparison with state-of-the-art methods. Some visualizations are shown in Figure 3.13.

- **SemanticPOSS** (*Pan et al.*, 2020a) contains about 3000 scans at the Peking University, which are divided into 6 equal subsets. For the experiments, we use the 3rd subset for evaluation and the others for training as in *Pan et al.* (2020a). There are usually more moving and small objects in the campus scenarios, making it more difficult compared to urban environments.

- **nuScenes** (*Caesar et al.*, 2020) includes about 40000 annotated scans captured in 900 scenes, where 750 scenes are for training and the others for validation. As recommended[1], we merge similar classes and remove rare classes.

- **Pandaset**[2] contains about 16000 LiDAR scans at 2 routes in Silicon Valley. Two LiDAR sensors have been used for recording the data, a spinning LiDAR and a solid-state LiDAR. For the experiments, the data from the spinning LiDAR is used. We use 30% of the data for evaluation and the rest for training. Similar to the nuScenes dataset, we merge similar classes and remove rare classes.

3.4.3 Evaluation Metric

As for the evaluation metric, we calculate the standard mean intersection over union (mIoU) (*Everingham et al.*, 2015) over all classes:

$$\text{mIoU} = \frac{1}{N} \sum_{n=1}^{N} \frac{\text{TP}_n}{\text{TP}_n + \text{FP}_n + \text{FN}_n} \tag{3.15}$$

where TP_n, FP_n, and FN_n denote the numbers of true positive, false positive, and false negative predictions for class n, respectively. N is the number of classes.

3.5 Motion Forecasting

Motion forecasting in autonomous driving involves predicting the future trajectories of surrounding agents, such as vehicles, pedestrians, and cyclists. It is essential for ensuring safety, optimizing routes, understanding agent intentions, and handling complex driving scenarios. Techniques used for accurate motion forecasting include spatial and temporal modeling. In essence, motion forecasting is pivotal for autonomous vehicles to anticipate and adeptly respond to evolving traffic situations.

3.5.1 Map Representation

To encode context information, the High-definition map (HD map), which is a map representation targeting at autonomous driving, is usually converted into different map representations according to the requirement. There are usually two main map representations, grid map and lane map which are shown in Figure 3.14.

3.5.1.1 Grid Map (Rasterized Representation)

In the domain of motion forecasting for autonomous driving, the grid map representation plays a pivotal role by converting intricate high-definition (HD) maps into a structured, discretized format. This involves dividing the continuous spatial environment into a matrix of uniform cells or grids. Each individual cell or grid encodes crucial information derived from the HD map, capturing details such as road boundaries, pedestrian pathways, intersections, and even dynamic entities like vehicles,

[1] The nuScenes LiDAR segmentation task is available at https://www.nuscenes.org/lidar-segmentation.
[2] https://scale.com/open-datasets/pandaset.

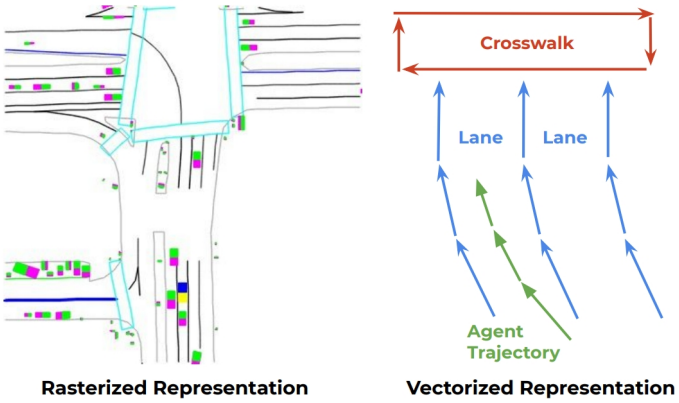


Figure 3.14: Map representations in motion forecasting. This image is from *Gao et al.* (2020).

cyclists, or pedestrians in motion. The primary advantage of this discretization process is the creation of a standardized and scalable input, which can be easily consumed by machine learning models and prediction algorithms. Moreover, the fusion of both static information from HD maps and real-time sensor data within this grid structure provides a comprehensive, momentary snapshot of the environment, essential for safe and informed decision-making by autonomous systems. However, this grid-based approach does come with its own challenges. Decisions regarding grid resolution can impact the quality and accuracy of predictions; finer grids capture more detail but at the cost of computational efficiency, while coarser grids might overlook crucial nuances. Furthermore, while grid maps are adept at capturing a lot of the scene's details, they might not always convey the hierarchical and semantic understanding of road structures as efficiently as more specialized map formats, such as lane maps, which are tailored to represent roadways and their associated attributes explicitly.

3.5.1.2 Lane Map (Vectorized Representation)

In the domain of autonomous driving, the lane map stands as a pivotal tool designed to meticulously capture the nuances of road lanes and their intricate structures. Unlike broader mapping frameworks, a lane map zeros in on the specifics, charting out every curve, marking, and deviation of the lane structures, making it a crucial component for sophisticated driving systems. These maps provide comprehensive details of the roadways, showcasing lane boundaries, categorizing lanes based on their usage (e.g., turning lanes, merging lanes, bus lanes), and embedding essential metadata such as lane width, type, direction of permissible traffic flow, and any special regulations or guidelines associated with them.

Moreover, they encapsulate the hierarchical framework of road systems, deftly distinguishing between primary arterials, secondary lanes, and tertiary byways. Such granularity becomes especially crucial at complex intersections and in scenarios with multilane configurations. An integrated view of lane connections, bifurcations, and potential merge points is presented, aiding autonomous systems in understanding permissible driving actions. When harnessed for motion forecasting, the rich tapestry of information presented by lane maps plays a decisive role. Vehicles, relying on this detailed map, can predict their trajectories with increased accuracy, understanding their relative position in conjunction with the broader lane layout. This not only facilitates more informed decision-making for autonomous vehicles but also amplifies safety measures, ensuring they adhere to lane protocols

and anticipate the movement of other vehicles with respect to the lane blueprint.

3.5.2 Datasets

This thesis includes both mapless trajectory prediction and map-assisted trajectory prediction.

3.5.2.1 Mapless trajectory prediction

We evaluate our method on three common trajectory forecasting datasets and follow the common evaluation setting that 8 frames are used as observation and the following 12 frames need to be forecast. All trajectories are provided in 2D coordinates.

The **ETH** (*Pellegrini et al.*, 2009) and **UCY** (*Lerner et al.*, 2007) datasets aim at pedestrian trajectory forecasting. ETH contains 2 top view scenes with 750 trajectories and UCY contains 3 scenes close to the top view with 786 trajectories. The data processing is done as *Gupta et al.* (2018). The sampling rate is 2.5 frames per second (one frame per 0.4s), which means that the model observes 3.2 seconds and predicts the trajectories for the next 4.8 seconds.

The **Stanford Drone Dataset** (*Robicquet et al.*, 2016) includes a series of videos captured by a hovering drone from 8 different college campuses with more than 10,000 trajectories. The dataset is much larger than ETH and UCY and includes other objects like bikes or cars apart from pedestrians. The frame rate is 2.5 frames per second, which is the same as for ETH and UCY.

3.5.2.2 Map-assist trajectory prediction

We evaluate our method on a widely used dataset for trajectory forecasting: nuScenes (*Caesar et al.*, 2020). It provides high-definition (HD) maps associated with the trajectory data. The nuScenes dataset includes 245,414 trajectory instances across 1,000 different scenes. Each trajectory instance lasts for 8 seconds sampled at 2Hz. The trajectory forecasting task defined in the nuScenes dataset requires forecasting the future trajectory of the ego vehicle for 6 seconds given the first 2 seconds as the observed trajectory.

3.5.3 Evaluation Metrics

We report the widely used minimum average displacement error ADE_K and the final displacement error FDE_K of K forecast trajectories with respect to the ground-truth trajectory $\mathcal{T}^f$:

$$ADE_K(\hat{\mathcal{T}}^f_{1:K}, \mathcal{T}^f) = \min_k \left\{ \frac{1}{T_f} \sum_{t=1}^{T_f} ||x_t - \hat{x}^k_t||_2 \right\}, \tag{3.16}$$

$$FDE_K(\hat{\mathcal{T}}^f_{1:K}, \mathcal{T}^f) = \min_k ||x_{T_f} - \hat{x}^k_{T_f}||_2. \tag{3.17}$$

We also report the Miss Rate for different values of K and the Off Road Rate for the nuScenes dataset (*Caesar et al.*, 2020). The former metric is defined as the proportion of misses over all agents. If the maximum L2 distance between the best prediction and the ground truth trajectory is larger than 2m, this prediction is defined as a miss. Similarly, if a prediction does not entirely lie in the drivable area, it is defined as off road. The Off Road Rate is defined as the proportion of off road predictions over all agents.

CHAPTER 4

Multi-scale Interaction for Real-time LiDAR Data Segmentation on an Embedded Platform

Individual Contribution

The following chapter is based on the publication:

Multi-scale Interaction for Real-time LiDAR Data Segmentation on an Embedded Platform
ShiJie Li, Xieyuanli Chen, Yun Liu, Dengxin Dai, Cyrill Stachniss, Juergen Gall
IEEE Robotics and Automation Letters (RA-L), 2021.
doi:10.1109/LRA.2021.3132059.

Below we will summarize the contributions of each author.

- **Shijie Li**
  Shijie proposed the main idea of the paper and implemented the whole method. Shijie performed the experiments and conducted the qualitative and quantitative analysis. Shijie also wrote most of the paper.
- **Xieyuanli Chen, Yun Liu**
  Both contributed to the idea and writing.
- **Dengxin Dai, Cyrill Stachniss**
  Both contributed with supervision, discussions, and writing.
- **Juergen Gall**
  Juergen supervised the project. He contributed with discussion and writing.

In this chapter, we propose a projection-based method, called Multi-scale Interaction Network (MINet), which is very efficient and accurate. The network uses multiple paths with different scales and balances the computational resources between the scales. Additional dense interactions between the scales avoid redundant computations and make the network highly efficient. The proposed network outperforms point-based, image-based, and projection-based methods in terms of accuracy, number of parameters, and runtime. Moreover, the network processes more than 24 scans, captured by a high-resolution LiDAR sensor with 64 beams, per second on an embedded platform, which is higher than the framerate of the sensor. The network is therefore suitable for robotics applications.

Contents

4.1 Introduction

Environment perception and understanding are key to realize self-driving vehicles and robots. For the full-view perception of the environment, autonomously driving vehicles are usually equipped with multi-sensor systems, among which light detection and ranging (LiDAR) sensors play a key role due to their precise distance measurements. The large point clouds that are generated by the LiDAR sensors, however, need to be interpreted in order to understand the environment.

Although convolution neural networks (CNNs) perform well for semantic image segmentation (*Chen et al.*, 2017; *Zhao et al.*, 2017; *Howard et al.*, 2019; *Ma et al.*, 2018), they cannot be applied directly to 3D point clouds. This is because standard convolutions require a regular grid structure, whereas a raw point cloud is an unordered structure. To address this problem, some methods (*Qi et al.*, 2017b,d; *Wu et al.*, 2019c) directly process point clouds using some spatial aggregation operations like grouping and gathering. Although these methods work well in indoor scenarios, it is difficult to apply them to large outdoor scenarios since the computational cost of the aggregation operation increases with the number of points. Another issue is that these methods are inefficient on embedded platforms since they use operations that cannot be efficiently mapped on embedded hardware like Jetson AGX using TensorRT. However, runtime efficiency is of vital importance for real-world applications, especially for autonomously driving vehicles and robots.

Wu *et al.*(*Wu et al.*, 2018b, 2019b) thus proposed to represent point clouds produced by a LiDAR sensor as an ordered projection map, such that CNNs can then be applied. However, projected LiDAR data and RGB images are different modalities and applying directly 2D image-based methods does not yield a high segmentation accuracy. For this reason, some specific CNNs have been designed for LiDAR-based depth images, named as projection-based methods. Recent projection-based methods like *Milioto et al.* (2019b), however, are very large with more than 50M parameters, making them unsuitable for embedded platforms.

In this work, we therefore propose a lightweight projection-based model for semantic segmentation of LiDAR data that runs in real-time on an embedded platform. To this end, we revisit common multi-scale approaches like U-Net (*Ronneberger et al.*, 2015) that have one path for each scale, *i.e.*each scale is processed independently and then fused at the end of the network. These networks,

however, use the same operations for each path, which makes them either too expensive for embedded platforms or the accuracy is very low depending how complex the used operations are. In order to achieve a good balance between effectiveness and efficiency, we therefore adapt the computational operations for each path. While the top path extracts low-level clues, which can be easily detected with shallow layers operating on high-resolution feature maps, the bottom path extracts high-level semantic information, which requires more complex operations but on low-resolution feature maps. In order to avoid redundant computations across the paths, we propose a dense top-to-bottom interaction strategy where feature maps from a path are passed to all lower paths. We term the network, which is shown in Figure 4.1, Multi-scale Interaction Network (MINet).

In addition, we show that the accuracy can be increased if additional supervision is added. While this is consistent with *Zhao et al.* (2017); *Zhang et al.* (2018); *Liu et al.* (2019b, 2018), we demonstrate that it is important to use the right type of supervision for the right part of the network. In fact, we use semantic supervision only for the two top paths but not for the bottom path and edge supervision for the fusion of the multiple paths. The latter is important to obtain accurate segment boundaries after upsampling the paths with lower resolution. Finally, we process the multi-modal data consisting of 3D coordinates, remission, and depth information first independently and then fuse them in the feature space. This is in contrast to previous works for LiDAR data that just concatenate the modalities and therefore ignore that the characteristics of each modality are different.

In summary, our contributions include:

- We propose a multi-scale approach where the computational operations are balanced across the different scales and a top-to-bottom interaction strategy avoids redundant computations.
- We exploit different types of additional supervision to improve the accuracy without increasing the inference time.
- Different from previous methods, we process each modality independently and fuse them in the feature space, which improves the overall segmentation performance.
- By incorporating the above design decisions, we propose a lightweight projection-based model for semantic segmentation of LiDAR data that runs in real-time on an embedded platform.

The experimental results demonstrate that our method reduces the number of parameters by about 98% and is about 4× faster than the state-of-the-art projection-based method (*Milioto et al.*, 2019b), while achieving a higher accuracy. We also evaluate our model on an embedded platform and demonstrate that our method can be deployed for autonomous driving.

4.2 Multi-scale Interaction Network

The proposed Multi-scale Interaction Network (MINet) operates on projection maps generated from LiDAR point clouds. To associate a LiDAR point $\mathbf{a} = (x, y, z)$ to a pixel (u, v) in the projection map of size (h, w), we compute yaw by Equation 4.1 and pitch by Equation 4.2 and map it to pixel coordinates by translation and scaling (*Milioto et al.*, 2019b):

$$u = \frac{1}{2}[1 - \arctan2(y, x)\pi^{-1}]w, \tag{4.1}$$

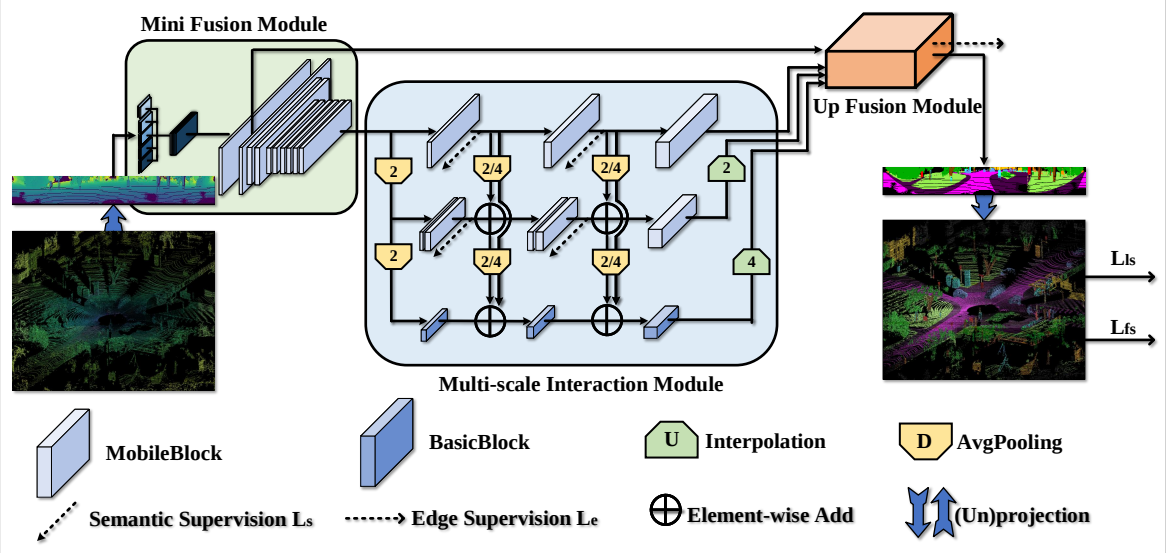


Figure 4.1: Illustration of the MINet architecture with three paths in the Multi-scale Interaction Module. The numbers 2 and 4 for interpolation (U) and average pooling (D) indicate the upsampling and downsampling factor. The dashed arrows indicate the supervision type. The detailed description of the architecture is given in Table 4.1 where the different blocks are illustrated in Figure 4.2 and the Up Fusion Module is illustrated in Figure 4.3.

$$v = [1 - (\arcsin(zd^{-1}) + o_{up})o^{-1}]h. \tag{4.2}$$

The vertical field-of-view of the LiDAR sensor is $o = o_{up} + o_{down}$, where o_{up} and o_{down} represent the above and below horizon of the field-of-view, respectively. $d = ||\mathbf{a}||$ denotes the depth of a point. After this transformation, we obtain a projection map of size $(h, w, 5)$ where the 5 channels correspond to the coordinates (x, y, z), the depth, and the remission of the corresponding 3D point. The remission value indicates the proportion of the light that is diffusely reflected. It provides therefore information of the surface, which is helpful for distinguishing different classes. While depth can be computed from the coordinates, the network operations do not compute the depth explicitly. Adding depth in addition to the coordinates thus improves the accuracy as we show in our experiments. Each channel is normalized by the mean and standard deviation computed over the training and validation set.

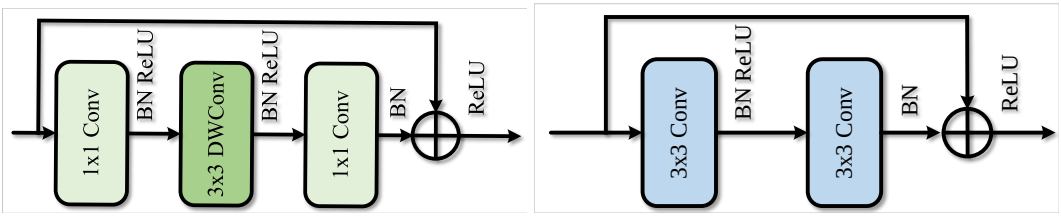


Figure 4.2: Illustration of the MobileBlock (left) and the BasicBlock (right). DWConv means depth-wise convolution.

The architecture of MINet is shown in Figure 4.1. The projection map is first processed by the Mini Fusion Module (MFM) (Section. 4.2.1) to fuse the multi-modal information in the feature space.

In the Multi-scale Interaction Module (MIM) (Section. 4.2.2), the data is processed at three different scales where the resolution is reduced by factor two for each path. As it is shown in Table 4.1, the computation differs for each path where we use two basic components, namely MobileBlock and BasicBlock. The MobileBlock (*Howard et al.*, 2017; *Sandler et al.*, 2018) utilizes depthwise convolutions and has thus fewer parameters, but its learning capacity is also limited. The BasicBlock (*He et al.*, 2016a) is stronger, but also more expensive. MobileBlock and BasicBlock are shown in Figure 4.2. We therefore balance the computational resources across the three paths as it is shown in Table 4.1. While we use the expensive BasicBlock for the bottom path with lowest resolution, we decrease the computational cost as the resolution increases using five MobileBlocks for the middle path and three for the top path. The connections from each path to lower paths avoid redundant computations at lower paths and make the network more efficient. Finally, the 2D predictions for the original resolution are produced by the Up Fusion Module (UFM) (Section. 4.2.1), which are then mapped back to the 3D space. In the remainder of this section, we describe each module of MINet.

Module	Operation	k	c	s	t	Output size
MFM	Conv2d	3	4	1	5	64×2048
	MobileBlock	3	20	1	1	64×2048
	MobileBlock	3	24	2	1	32×512
	MobileBlock	3	24	1	1	32×512
	MobileBlock	5	40	2	1	16×256
	MobileBlock	5	40	1	1	16×256
	MobileBlock	5	40	1	1	16×256
	MobileBlock	3	80	1	1	16×256
	MobileBlock	3	80	1	1	16×256
	MobileBlock	3	80	1	1	16×256
	MobileBlock	3	80	1	1	16×256
	Conv2d	1	32	0	1	16×256
MIM	MobileBlock	3	64	1	1	16×256
	MobileBlock	3	128	1	1	16×256
	MobileBlock	3	128	1	1	16×256
	MobileBlock	3	32	1	1	8×128
	MobileBlock	3	64	1	1	8×128
	MobileBlock	3	64	1	1	8×128
	MobileBlock	3	128	1	1	8×128
	MobileBlock	3	128	1	1	8×128
	BasicBlock	3	64	1	1	4×64
	BasicBlock	3	128	1	1	4×64
	BasicBlock	3	128	1	1	4×64
UFM	Conv2d	3	32	1	1	16×512
	Conv2d	3	32	1	1	64×2048
	Conv2d	1	32	1	1	64×2048

* Each module contains several components: Conv2d, MobileBlock, and BasicBlock. Conv2d denotes a convolutional layer followed by one batch normalization layer and ReLU activation. MobileBlock and BasicBlock are illustrated in Figure 4.2. Each operation has a kernel size k, stride s, and c output channels, repeated t times. The three sections of MIM denote the three paths.

Table 4.1: Instantiation of the proposed MINet.

No.	MFM	Interaction	UFM	mIoU
1		✓	✓	50.9
2	✓		✓	50.7
3	✓	✓		50.6
4	✓	✓	✓	51.8
5	w/o depth	✓	✓	49.6

Table 4.2: Impact of the three modules.

4.2.1 Mini Fusion Module (MFM)

Different from an RGB image, the projection map contains channels of different modalities. Previous projection-based segmentation methods (*Wu et al.*, 2018b, 2019b; *Milioto et al.*, 2019b) treat such different modalities equally, but we show that processing each channel independently using MFM is more efficient. Specifically, each channel of the multi-modal image is mapped to an independent feature space using five convolution blocks, including normalization and activation. This corresponds to the first row of Table 4.1. This step can be considered as a feature calibration step for each modality before fusing them. It needs to be noted that we also treat the x, y, and z coordinates separately. After the first five convolutional layers, these features are concatenated and fed into several MobileBlocks for fusing them. Since a small resolution leads to less computation, the information of the feature maps are gradually aggregated by average pooling.

4.2.2 Multi-scale Interaction Module (MIM)

After the fusion module, the data is processed by three paths where each path corresponds to a different scale as shown in Figure 4.1. From top to bottom, the resolution of the feature maps is decreased by factor two using average pooling and the receptive field is accordingly increased. For the top path, we use the highest resolution. Since processing high resolution feature maps is very expensive, we use only three MobileBlocks as shown in Table 4.1. The bottom path, which has the largest receptive field and lowest resolution, can offer more abstract semantic clues if we use more expensive operations. Hence, it uses three BasicBlocks. The middle path is a compromise between the top and bottom path and consists of five MobileBlocks. In our experiments, we show that increasing the computational operations as the resolution decreases leads to a higher efficiency compared to using the same blocks for all paths. While the number of parameters doubles compared to the proposed architecture if we use the BasicBlocks for all paths, the accuracy drops if only MobileBlocks are used.

A second important design choice is to allow interactions among the paths. Since the computational complexity of the paths increases for lower paths, we use a dense top-to-bottom fusion design for efficient multi-scale feature interaction. Especially, feature maps of the first and second path will be resized by average pooling and passed to all lower paths. To avoid a mismatch of the number of channels, the number of channels is increased gradually for each path and kept the same at each interaction position. Hence, no other operations are used to adjust the number of channels as shown in Table 4.1. Due to the interaction, the lower paths benefit from the features computed from higher paths. The lower paths can therefore focus on information that has not been extracted by higher paths due to limited computational resources.

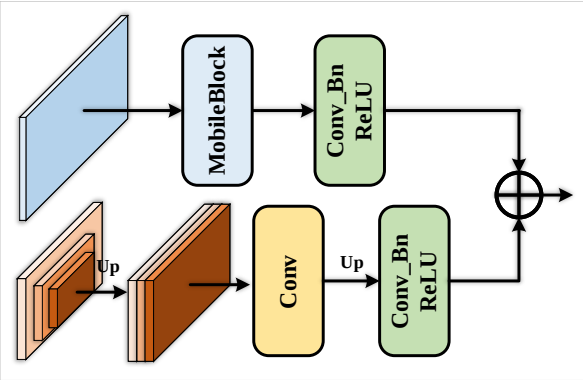


Figure 4.3: Illustration of the Up Fusion Module (UFM).

No.	MIM			UFM	mIoU
	Top	Middle	Bottom		
1	S	S		E	51.8
2		S		E	50.3
3	S			E	50.2
4	S	S			50.5
5				E	49.0
6					48.4
7	S	S	S	E	50.9
8	S	S		S	50.8
9	S	S		E(FL)	49.4

* "S" denotes semantic supervision. "E" denotes edge supervision. "(FL)" indicates that the focal loss is used for edge supervision.

Table 4.3: Impact of additional supervision.

4.2.3 Up Fusion Module (UFM)

To obtain the semantic labels of each pixel in the projection map, UFM shown in Figure 4.3 combines the features from different scales and upsamples them to the input resolution. In addition, features after the first MobileBlock of the Mini Fusion Module are used to recover detailed spatial information as shown in Figure 4.1. The lower part of Figure 4.3 shows the feature maps of the three different paths that are first resized to the same size, concatenated together, and fused by a 1×1 convolution. The fused feature maps are then upsampled to the original resolution and processed by a convolution block including a 3×3 convolution, batch normalization, and ReLU activation. The upper part shows the feature maps from the Mini Fusion Module that have already the original resolution. They are processed by a MobileBlock and a convolution block. Finally, the processed features from both modules are added together. Although the spatial information of the original feature maps already helps to sharpen segment boundaries, which can be fuzzy due to the upsampling, adding additional supervision for the segment boundaries emphasizes this effect as we will explain in the next section.

λ	0	0.01	0.1	1.0
mIoU	49.0	49.3	51.8	51.4

Table 4.4: Impact of λ in Equation 4.7.

4.2.4 Booster Training Strategy

Adding supervision to intermediate parts of a network (*Lee et al.*, 2015) has been shown to be useful for network optimization (*Zhao et al.*, 2017; *Zhang et al.*, 2018; *Liu et al.*, 2019b, 2018). In this work, we also use intermediate supervision, however, we propose two different types of supervision. Similar to balancing the computational resources across scales, it is very important to use the right supervision for the right part of the network.

We use the standard weighted cross-entropy loss as semantic supervision

$$\mathcal{L}_s = -\frac{1}{|I|}\sum_{i\in I}\sum_{n=1}^{N} w_n p_i^n \log(\hat{p}_i^n), \tag{4.3}$$

where N is the number of classes, $|I|$ is the total number of image pixels, p_i^n is the ground-truth semantic label for pixel i and class n ($p_i^n \in \{0,1\}$), and $\hat{p}_i^n$ is the predicted class probability. The weight w_n for class n is inversely proportional to its occurrence frequency as in *Milioto et al.* (2019b).

Besides, at the end of the network, we use the weighted cross-entropy loss $\mathcal{L}_s$ for the top and middle path as indicated by the dashed arrows in Figure 4.1. As we will show in the experiments, this intermediate supervision improves the training and boosts the accuracy. Adding this semantic supervision to the bottom path, however, does not help since the resolution of the lower path is too low, and the downsampling of the ground truth introduces too many artifacts.

As discussed in Section. 4.2.3, obtaining accurate segment boundaries after upsampling is an issue. Inspired by *Yu et al.* (2018b); *Kirillov et al.* (2017), we extract the semantic boundaries from the ground-truth labels and compare them to the semantic boundaries after upsampling. The semantic edge loss is then obtained by

$$\mathcal{L}_e = -\frac{1}{|I|}\sum_{i\in I}\left(e_i \log(\hat{e}_i) + (1-e_i)\log(1-\hat{e}_i)\right), \tag{4.4}$$

where e_i is the ground-truth edge label at pixel i ($e_i \in \{0,1\}$) and $\hat{e}_i$ is the predicted edge probability at pixel i.

Besides the weighted cross entropy loss, which we denote by $\mathcal{L}_{fs}$ and which is computed in the same way as $\mathcal{L}_s$, we use the Lovász-Softmax loss $\mathcal{L}_{ls}$ (*Berman et al.*, 2018a) at the end of the network, which maximizes the intersection-over-union (IoU) score:

$$\mathcal{L}_{ls} = \frac{1}{N}\sum_{n=1}^{N} \overline{\Delta_{J_n}}(m(n)), \tag{4.5}$$

$$m_i(n) = \begin{cases} 1-\hat{p}_i^n & \text{if } p_i^n = 1 \\ \hat{p}_i^n & \text{otherwise,} \end{cases} \tag{4.6}$$

where $\overline{\Delta_{J_n}}$ defines the Lovász extension of the Jaccard index, $\hat{p}_i^n \in [0,1]$ and $p_i^n \in \{0,1\}$ denote for class n at pixel i the predicted probability and ground-truth label, respectively. In summary, the

Top	Middle	Bottom	Param (M)	GFLOP	SPS	mIoU
3×MB	5×MB	3×BB	1.0	6.20	59	51.8
5×MB	5×MB	3×BB	1.1	6.43	52	50.1
7×MB	5×MB	3×BB	1.1	6.62	51	49.7
9×MB	5×MB	3×BB	1.2	7.77	48	48.9
3×MB	3×MB	3×BB	1.0	6.20	60	50.8
3×MB	7×MB	3×BB	1.0	6.29	55	49.0
3×MB	9×MB	3×BB	1.2	6.57	51	50.2
3×MB	5×MB	5×BB	1.4	6.43	54	50.3
3×MB	5×MB	7×BB	1.8	6.62	52	51.5
3×MB	5×MB	9×BB	2.4	6.92	50	49.8
3×MB	3×MB	3×MB	0.6	6.00	61	49.7
5×MB	5×MB	5×MB	0.6	6.30	58	50.2
3×BB	3×BB	3×BB	2.0	11.04	33	51.2

* "MB" denotes MobileBlock. "BB" denotes BasicBlock. "SPS" represents the number of processed scans per second.

Table 4.5: Ablation study for different blocks of each path.

combined loss is given by

$$\mathcal{L} = \mathcal{L}_{fs} + \mathcal{L}_{ls} + \mathcal{L}_{e} + \lambda \sum_{s} \mathcal{L}_{s} \tag{4.7}$$

where $\lambda = 0.1$ and $\mathcal{L}_s$ are the loss functions for the top and middle path. The arrows in Figure 4.1 show where each type of supervision is applied.

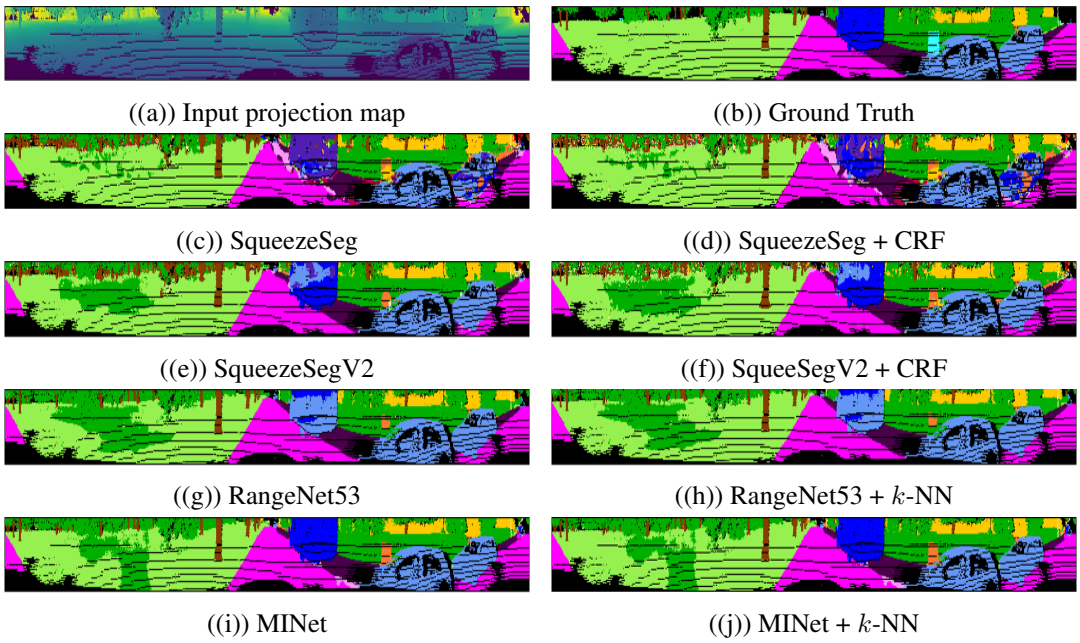
4.3 Experiments

4.3.1 Experimental Settings

We use two challenging datasets to evaluate our method, namely SemanticKITTI (*Behley et al.*, 2019d, 2021b) and SemanticPOSS (*Pan et al.*, 2020a). Based on the KITTI Odometry Benchmark (*Geiger et al.*, 2012b), SemanticKITTI provides a semantic label for each point in all scans. It includes over 43,000 scans from 21 sequences, among which sequences 00 to 10 with over 21,000 scans are available for training and the remaining scans from sequences 11 to 21 are used for testing. Sequence 08 is used as the validation set, and we train our approach on the remaining training set. We report the results on the validation set for the ablation study. For the test set, which we use to compare to the state-of-the-art, the ground-truth is withheld and the results are evaluated by an on-line server.

SemanticPOSS is a smaller dataset with 2988 LiDAR scans, captured at the Peking University. The point clouds of SemanticPOSS are more sparse compared to SemanticKITTI due to the lower resolution of the LiDAR sensor. SemanticPOSS is split into six subsets equally, among which we use the 3^{rd} subset for validation and the others for training. We report the results on the validation set. Since these two datasets differ in size, LiDAR sensor, and environment, they provide an ideal testbed for evaluating the proposed approach. As for the evaluation metric, we calculate the standard mean intersection over union (mIoU) (*Everingham et al.*, 2015) over all classes:

$$\text{mIoU} = \frac{1}{N} \sum_{n=1}^{N} \frac{\text{TP}_n}{\text{TP}_n + \text{FP}_n + \text{FN}_n} \tag{4.8}$$



((a)) Input projection map ((b)) Ground Truth

((c)) SqueezeSeg ((d)) SqueezeSeg + CRF

((e)) SqueezeSegV2 ((f)) SqueeSegV2 + CRF

((g)) RangeNet53 ((h)) RangeNet53 + k-NN

((i)) MINet ((j)) MINet + k-NN

Figure 4.4: Qualitative results. For the input image, we only show the depth.

where TP_n, FP_n, and FN_n denote the numbers of true positive, false positive, and false negative predictions for class n, respectively. N is the number of classes.

4.3.2 Ablation Study

4.3.2.1 Modules

As illustrated in Figure 4.1, our network consists of three modules. In Table 4.2 we evaluate some design choices for the Mini Fusion Module (MFM) (Section. 4.2.1), the Multi-scale Interaction Module (MIM) (Section. 4.2.2), and the Up Fusion Module (UFM) (Section. 4.2.3). While the proposed approach achieves 51.8% mIoU (Row 4), we evaluate the impact of processing each modality separately before fusing them in the features space in the first row. To keep the number of parameters the same, we use 3×3 convolutions to process the input multi-modal image instead of processing each modality separately in MFM. In this case, the accuracy drops by 0.9% (Row 1). This shows the benefit of processing each modality separately at the beginning. In the last row, we also report the result when we omit the depth from the input. In this case, the accuracy drops to 2.2%. In Figure 4.1, we have connections between the three paths. If we remove the top-to-down interactions, the accuracy is reduced by 1.1% (Row 2). This demonstrates the benefit of allowing interactions between the multi-scale features. If the multi-resolution features are just resized, concatenated, and processed by convolutional layers, instead of using UFM, the accuracy is reduced by 1.2% (Row 3).

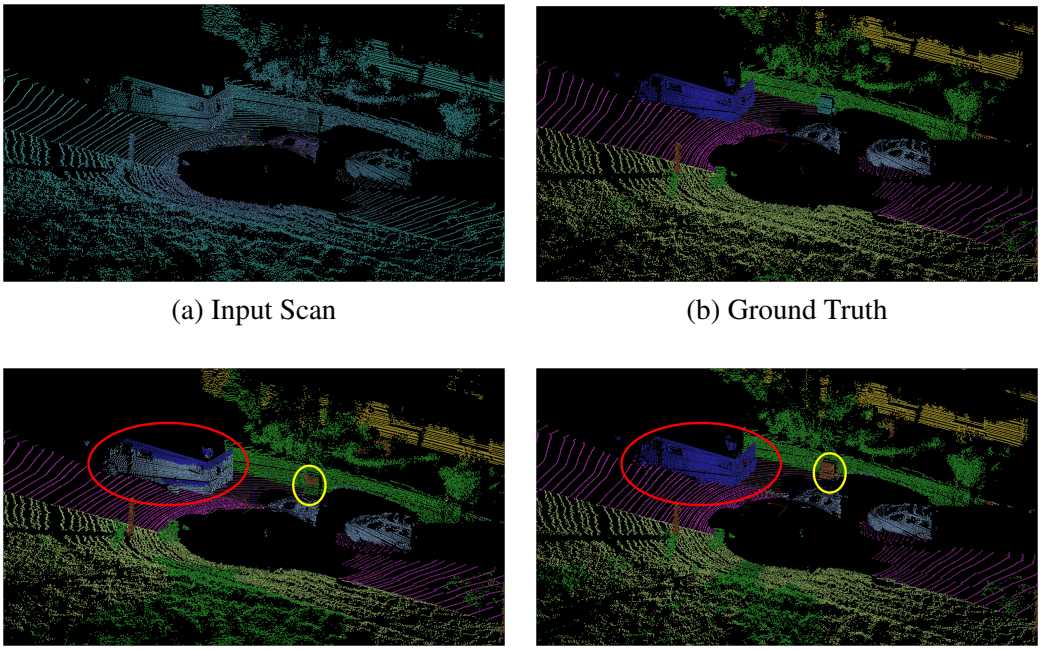
4.3.2.2 Supervision Setting

As discussed in Section. 4.2.4, we use additional supervision for MIM and UFM. For MIM, we use the semantic labels (S) to add the loss Equation 4.3 to the top and middle path. For UFM, we add the loss Equation 4.4 for the segment boundaries (E). In Table 4.3, we report the mIoU for different

Methods [height=0.63in,width=1.35in] Classes	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	SPS	Param (M)	mIoU
Pointnet (*Qi et al.*, 2017b)	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7	2	3.0	14.6
Pointnet++ (*Qi et al.*, 2017d)	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9	0.1	6.0	20.1
SPGraph (*Landrieu and Simonovsky*, 2018)	68.3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5	0.2	**0.3**	20.0
SPLATNet (*Su et al.*, 2018)	66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0	1	0.8	22.8
TangentConv (*Tatarchenko et al.*, 2018)	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2	0.3	0.4	35.9
P^2Net (*Li et al.*, 2020c)	85.6	20.4	14.4	14.4	11.5	16.9	24.9	5.9	87.8	47.5	67.3	7.3	77.9	43.4	72.5	36.5	60.8	22.8	38.2	12	6.0	39.8
RandLA-Net (*Hu et al.*, 2019)	**94.2**	26.0	25.8	**40.1**	**38.9**	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7	16	2.1	53.9
PolarNet (*Zhang et al.*, 2020b)	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	**90.0**	**61.3**	**84.0**	**65.5**	**67.8**	**51.8**	57.5	8	16.6	54.3
DeepLabV3+(*Chen et al.*, 2018)	78.4	13.6	9.5	9.5	10.4	17.5	22.0	0.4	88.5	54.5	66.7	9.7	77.9	39.1	72.0	39.9	60.0	23.4	36.1	39	59.4	38.4
PSPNet(*Li et al.*, 2018)	79.6	25.0	26.4	17.5	24.0	34.1	28.4	7.3	90.2	58.2	70.2	19.9	79.7	43.5	74.2	43.2	61.2	23.1	37.5	31	49.1	44.4
BiSeNet(*Yu et al.*, 2018a)	76.0	13.4	11.9	18.3	7.6	16.4	26.0	0.5	87.6	49.9	64.2	6.5	74.7	34.7	69.7	36.8	58.0	19.6	32.3	50	50.9	37.1
DenseASPP(*Yang et al.*, 2018)	78.1	20.5	18.2	20.0	16.6	27.8	28.9	5.7	88.5	53.3	67.5	9.3	76.3	39.6	70.0	36.8	57.7	15.9	32.4	20	23.4	40.2
SqueezeSeg (*Wu et al.*, 2018b)	68.8	16.0	4.1	3.3	3.6	12.9	13.1	0.9	85.4	26.9	54.3	4.5	57.4	29.0	60.0	24.3	53.7	17.5	24.5	**66**	1.0	29.5
SqueezeSeg + CRF (*Wu et al.*, 2018b)	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	55	1.0	30.8
SqueezeSegV2 (*Wu et al.*, 2019b)	81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3	50	1.0	39.7
SqueezeSegV2 + CRF (*Wu et al.*, 2019b)	82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0	40	1.0	39.6
RangeNet21 (*Milioto et al.*, 2019b)	85.4	26.2	26.5	18.6	15.6	31.8	33.6	4.0	91.4	57.0	74.0	26.4	81.9	52.3	77.6	48.4	63.6	36.0	50.0	20	25.0	47.4
RangeNet53 (*Milioto et al.*, 2019b)	86.4	24.5	32.7	25.5	22.6	36.2	33.6	4.7	**91.8**	64.8	74.6	**27.9**	84.1	55.0	78.3	50.1	64.0	38.9	52.2	13	50.4	49.9
RangeNet53 + k-NN (*Milioto et al.*, 2019b)	91.4	25.7	**34.4**	25.7	23.0	38.3	38.8	4.8	**91.8**	**65.0**	**75.2**	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9	12	50.4	52.2
MINet	85.2	38.2	32.1	29.3	23.1	47.6	46.8	24.5	90.5	58.8	72.1	25.9	82.2	49.5	78.8	52.5	65.4	37.7	55.5	59	1.0	52.4
MINet + k-NN	90.1	**41.8**	34.0	29.9	23.6	**51.4**	**52.4**	**25.0**	90.5	59.0	72.6	25.8	85.6	52.3	81.1	58.1	66.1	49.0	**59.9**	47	1.0	**55.2**

* The proposed MINet performs well for small objects.

Table 4.6: Evaluation results on the SemanticKITTI test set for point-based, image-based, and projection-based methods.



(a) Input Scan (b) Ground Truth

(c) RangeNet53 + k-NN (d) MINet + k-NN

Figure 4.5: Qualitative results. RangeNet segments most points of the truck as a car, while MINet segments the truck correctly (red circle). In some cases, both approaches fail (yellow circle). Although MINet segments the object correctly, it misclassifies it.

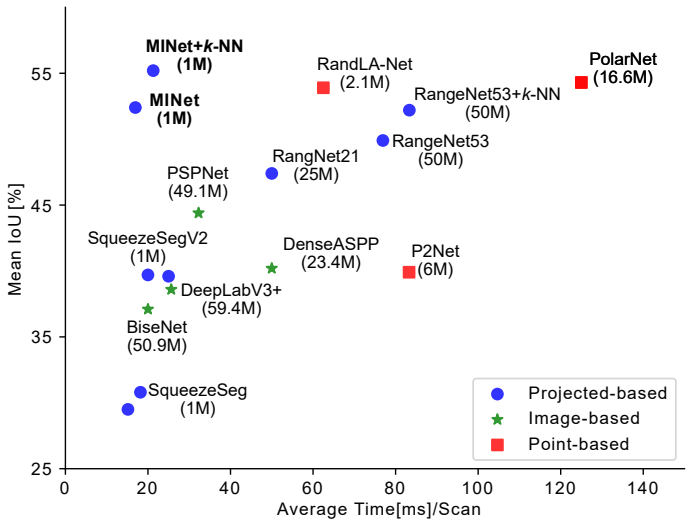
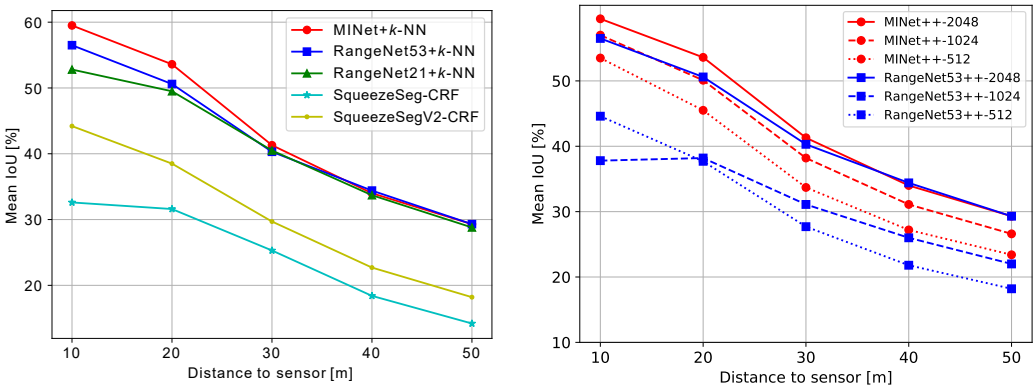


Figure 4.6: Visualization of Table 4.6. We omit point-based methods that need more than 140ms per scan. The proposed MINet is not only the most accurate method, it also achieves the best trade-off between accuracy and runtime.

settings. The best setting is achieved by adding semantic supervision to the top and middle path in MIM and applying edge supervision to UFM (Row 1). Removing any of this additional supervision



(a) Distance (b) Resolution

Figure 4.7: Accuracy of different projection-based methods based on the distance to the sensor (a) and the impact of resolution.

	person	rider	car	trunk	plants	traffic sign	pole	building	fence	bike	road	mIoU
SqueezeSegV1(*Wu et al.*, 2018b)	5.5	0.0	8.7	3.4	39.1	2.4	2.5	34.5	7.6	18.4	62.5	16.8
SqueezeSegV1(*Wu et al.*, 2018b) + CRF	14.2	1.4	11.6	18.1	5.9	11.1	1.9	37.9	5.6	18.9	78.7	18.7
SqueezeSegV2(*Wu et al.*, 2019b)	18.4	11.2	34.9	15.8	56.3	11.0	4.5	47.0	25.5	32.4	71.3	29.8
SqueezeSegV2(*Wu et al.*, 2019b) + CRF	**23.9**	**22.6**	29.7	15.3	37.3	11.1	**5.3**	45.9	18.2	34.7	**73.4**	28.9
RangeNet53(*Milioto et al.*, 2019b)	10.0	6.2	33.4	7.3	54.2	5.5	2.6	49.9	18.4	28.6	63.5	25.4
RangeNet53 + k-NN	14.2	8.2	35.4	9.2	58.1	6.8	2.8	55.5	**28.8**	32.2	66.3	28.9
MINet	13.3	11.3	34.0	18.8	62.9	11.8	4.1	55.5	20.4	34.7	69.2	30.5
MINet + k-NN	20.1	15.1	**36.0**	**23.4**	**67.4**	**15.5**	5.1	**61.6**	28.2	**40.2**	72.9	**35.1**

Table 4.7: Evaluation results on the SemanticPOSS dataset.

leads to an accuracy loss by more than 1.3% (Row 2-4). If the additional supervision is only used for UFM, the accuracy even decreases further (Row 5). If we do not use any additional supervision, the accuracy is lowest and 3.4% below the proposed setting (Row 6). While we removed so far additional supervision, the last two rows in the table show results when we add or change the type of supervision. If we add additional supervision to the bottom path, the accuracy decreases by 0.9%. This is due to the large difference between the ground-truth resolution and the resolution of the bottom path, which results in sampling artifacts that have a negative impact. Instead of using the edge loss Equation 4.4 for UFM, we also replaced it by the semantic loss that is used for MIM. While adding the semantic loss (Row 8) is better than using no additional loss for UFM (Row 4), the edge loss (Row 1) achieves a 1% higher accuracy than the semantic loss. This is expected since the purpose of the edge loss is to improve the segment boundaries after the upscaling, which is done by the Up Fusion Module. We also investigated what happens if the focal loss (*Lin et al.*, 2017) is used for the edge loss instead of Equation 4.4 (Row 9). In this case, the accuracy decreases.

4.3.2.3 Impact of λ

Our loss function Equation 4.7 contains only one hyper-parameter, namely λ. We evaluate the impact of λ in Table 4.4. The setting $\lambda = 0$ corresponds to row 5 of Table 4.3 where no additional supervision is added to the paths. Setting λ between 0.1 and 1.0 performs well.

Methods	Resolutions	GFLOPs	SPS	mIoU
RangeNet53 (*Milioto et al.*, 2019b)	64 × 2048	360.5	7	49.9
	64 × 1024	180.3	11	45.4
	64 × 512	90.1	22	39.3
RangeNet53 + k-NN (*Milioto et al.*, 2019b)	64 × 2048	360.5	5	52.2
	64 × 1024	180.3	8	48.0
	64 × 512	90.1	13	41.9
MINet	64 × 2048	6.2	24	52.4
	64 × 1024	3.2	47	49.1
	64 × 512	1.7	80	45.0
MINet + k-NN	64 × 2048	6.2	13	55.2
	64 × 1024	3.2	18	52.4
	64 × 512	1.7	21	48.5

* The number of GFLOPs does not include the post-processing.

Table 4.8: Performance on an embedded platform (Jetson AGX).

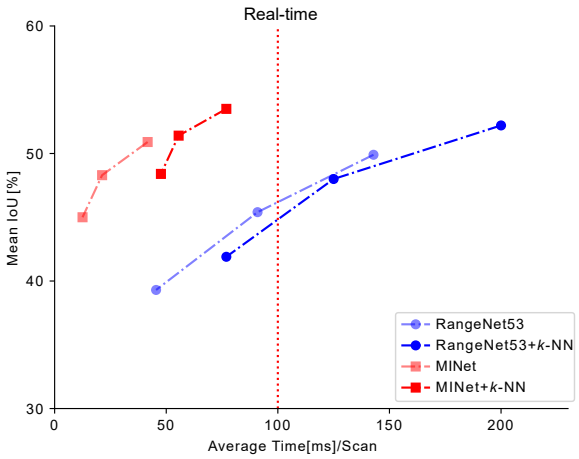


Figure 4.8: Visualization of Table 4.8. For each method, the three points on its curve correspond to the three different input resolutions 64×512, 64×1024, and 64×2048 from left to right. MINet runs at a real-time speed on a Jetson AGX with full resolution and post-processing.

4.3.2.4 Path Settings

As shown in Table 4.1, we balance the computational resources across the three paths where we increase the complexity as the resolution decreases. In rows 1-10 of Table 4.5, we report the results when we vary the number of blocks for the top, middle, and bottom path. The results show that increasing the parameters only for one path does not result in an improvement. For instance, using 9 instead of 3 MobileBlocks for the top path (Row 4) decreases the accuracy by 2.9%. This shows that a good computational balance between the paths is required. In rows 11-13, we report the results when we use the same operations for all paths, *i.e.*either 3 or 5 MobileBlocks or 3 BasicBlocks. Using only MobileBlocks reduces the number of parameters, but it improves the runtime only slightly and this is only the case for 3 MobileBlocks. This, however, comes at a substantially lower accuracy. In terms of runtime and accuracy, the proposed setting provides a much better trade-off. If the computationally expensive BasicBlocks are used for all paths, the number of parameters and runtime nearly doubles

while the accuracy is nearly the same. This shows that using the same operations for all resolutions is highly inefficient and that the proposed approach achieves a good balance between efficiency and effectiveness.

4.3.3 Comparison with other Methods

We first compare the proposed approach (MINet) with other methods on the SemanticKITTI test set in terms of both accuracy and efficiency. For a fair comparison, all methods including image-based methods are trained from scratch. The results are shown in Table 4.6. In the first rows, we show the results for point-based methods. Most of the approaches are very slow and cannot process more than 2 scans per second since spatial aggregation operations are usually very time-consuming for large point clouds. The very recent works (*Li et al.*, 2020c; *Hu et al.*, 2019; *Zhang et al.*, 2020b) are faster and process up to 16 scans per second, but it is difficult to deploy these networks on embedded systems due to their complex operations. The highest accuracy is achieved by *Zhang et al.* (2020b), but the network is very large with over 16M parameters. Our proposed approach outperforms all point-based methods in terms of runtime, number of parameters, and accuracy.

As for image-based methods, we use four widely used methods, namely PSPNet (*Zhao et al.*, 2017), DeepLabV3+ (*Chen et al.*, 2018), DenseASPP (*Yang et al.*, 2018), and the lightweight model BiseNet (*Yu et al.*, 2018a). We adjust the input channels so that these methods can be applied to the projection map. While these methods are faster than point-based methods, they have by far more parameters and the accuracy is significantly lower. This shows that projected LiDAR data cannot be directly processed by image-based segmentation methods since the modality differs from RGB images.

We also compare our approach to other projection-based methods. While SqueezeSeg (*Wu et al.*, 2018b, 2019b) has the same amount of parameters and depending on the setting runs slightly faster, the accuracy is very low. The state-of-the-art projection-based method RangeNet (*Milioto et al.*, 2019b) outperforms SqueezeSeg in terms of accuracy, but this is achieved by increasing the number of parameters to over 50M and decreasing the runtime. Our approach is much more efficient. It uses only 2% of the number of parameters compared to RangeNet53 and it is about 4× faster while achieving a higher accuracy. Figure 4.6 visualizes the accuracy and runtime of the methods of Table 4.6 and shows the effectiveness and efficiency of the proposed approach.

We also evaluate our method on SemanticPOSS (*Pan et al.*, 2020a) and compare our approach to other methods in Table 4.7. Since the dataset is smaller and the point clouds are more sparse compared to SemanticKitti, the mIoU is lower for all methods. However, our approach still outperforms other methods with a large margin. This proves the effectiveness of our method.

We also show the performance variance according to the distance to the sensor in Figure 4.7. Some qualitative results are shown in Figure 4.4 and Figure 4.5.

4.3.4 Performance on an Embedded Platform

We finally compare our method with RangeNet on an embedded platform. Here, we use a Jetson AGX that is an AI module for embedded systems, as it is usually used for autonomous driving. We optimize our method and RangeNet using TensorRT. The results are summarized in Table 4.8 and visualized in Figure 4.8. Since the input resolution can be decreased to reduce the runtime, we report the results for three different input resolutions. We can see that at each resolution, the

proposed MINet outperforms RangeNet53 (*Milioto et al.*, 2019b) with or without post-processing. Furthermore, MINet is also much faster than RangeNet53. Specifically, MINet is about 4× faster than RangeNet53 without post-processing and 2× faster with post-processing. Such high efficiency makes MINet quite suitable for robotics applications. Even with full resolution and post-processing, MINet runs at real-time since the LiDAR scan frequency is 10Hz. Compared to Table 4.6 where the runtime is measured on a workstation with a single Quadro P6000, the post-processing has a higher impact on the runtime for the embedded platform since the post-processing is not optimized by TensorRT. Moreover, the post-processing is applied to the point cloud, so it cannot benefit from reducing the resolution of the projection map.

4.4 Summary

In this work, we proposed a novel lightweight projection-based method, called Multi-scale Interaction Network (MINet), for semantic segmentation of LiDAR data. The network is highly efficient and runs in real-time on GPUs and embedded platforms. It outperforms point-based, image-based, and projection-based methods in terms of accuracy, number of parameters, and runtime. This is achieved by using a multi-scale approach where the computational resources are balanced between the scales and by introducing interactions between the scales. By processing the modalities separately before fusing them and adding additional different types of supervision, we could further improve the accuracy without decreasing the runtime. Compared to the state-of-the-art projection-based method RangeNet, MINet reduces the number of parameters by 98% and is 4× faster while achieving a higher accuracy. Since MINet processes more than 24 scans per second on an embedded platform, it can be used for autonomous vehicles and robots. Our method also achieves good performance on other tasks, like moving object segmentation (*Chen et al.*, 2021b). Projection-based methods, however, have some limitations. For instance, it is not straightforward to integrate temporal information from multiple views. Finally, we expect that the design principles of the network are also valuable for other tasks like 3D car detection. The source code is available at `https://github.com/sj-li/MINet`.

CHAPTER 5

Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data

Individual Contribution

The following chapter is based on the publication:

Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data
Xieyuanli Chen, Shijie Li, Benedikt Mersch, Louis Wiesmann, Juergen Gall, Jens Behley, Cyrill Stachniss
IEEE Robotics and Automation Letters (RA-L), 2021.
doi:10.1109/LRA.2021.3093567

Below we will summarize the contributions of each author.

- **Xieyuanli Chen**
  Xieyuanli proposed the main idea of the paper and implemented the whole method. Xieyuanli performed the experiments of localization and mapping with the proposed method and conducted qualitative and quantitative analysis. Xieyuanli also wrote most of the paper.

- **Shijie Li**
  Shijie participated in the discussion and contributed to the main idea. Shijie conducted the training of the proposed method on the SemanticKITTI dataset. Shijie also contributes to the writing.

- **Benedikt Mersch, Louis Wiesmann**
  Both contributed to writing.

- **Juergen Gall, Jens Behley**
  Both contributed to discussions and writing.

- **Cyrill Stachniss**
  Cyrill supervised the project. He contributed with discussion and writing.

In the previous chapter, we have proposed a highly efficient projection-based architecture for LiDAR point cloud semantic segmentation. Apart from semantic information, moving information is

also very important in autonomous driving perception modules which is usually ignored in previous research. The ability to detect and segment moving objects in a scene is essential for building consistent maps, making future state predictions, avoiding collisions, and planning. In this task, instead of assigning each point a semantic label, we want to distinguish the moving status of each point.

In this chapter, we address the problem of moving object segmentation from 3D LiDAR scans. We propose a novel approach that pushes the current state of the art in LiDAR-only moving object segmentation forward to provide relevant information for autonomous robots and other vehicles. Instead of segmenting the point cloud semantically, *i.e.*, predicting the semantic classes such as vehicles, pedestrians, buildings, roads, etc., our approach accurately segments the scene into moving and static objects, *i.e.*, distinguishing between moving cars vs. parked cars. Our proposed approach exploits sequential range images from a rotating 3D LiDAR sensor as an intermediate representation combined with a convolutional neural network and runs faster than the frame rate of the sensor. We compare our approach to several other state-of-the-art methods showing superior segmentation quality in urban environments. Additionally, we created a new benchmark for LiDAR-based moving object segmentation based on SemanticKITTI.

Contents

5.1 Introduction

The ability to identify which parts of the environment are static and which ones are moving is key to safe and reliable autonomous navigation. It supports the task of predicting the future state of the surroundings, collision avoidance, and planning. This knowledge can also improve and robustify pose estimation, sensor data registration, and simultaneous localization and mapping (SLAM). Thus, accurate and reliable moving object segmentation (MOS) in sensor data at frame rate is a crucial capability supporting most autonomous mobile systems. Depending on the application domain and chosen sensor setup, moving object segmentation can be a challenging task.

In this work, we address the problem of moving object segmentation in 3D LiDAR data at sensor frame rate in urban environments. Instead of detecting all *theoretically movable* objects such as vehicles or humans, we aim at separating the *actually moving* objects such as driving cars from

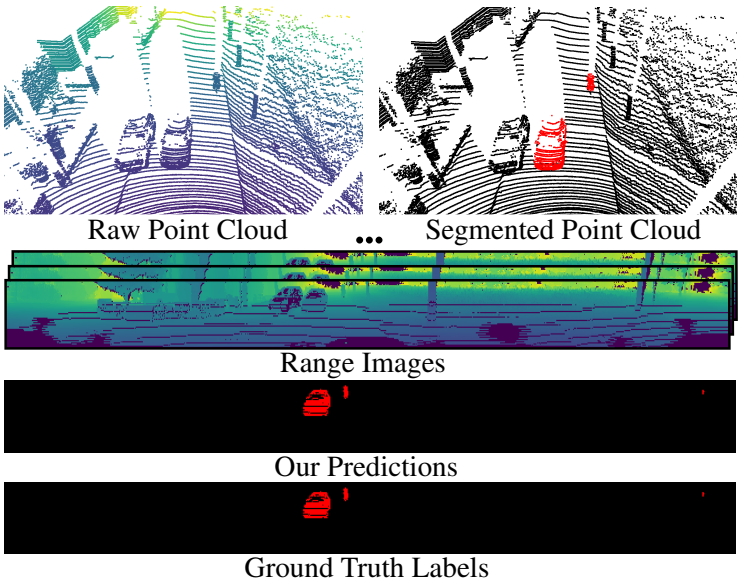


Figure 5.1: Moving object segmentation using our approach. Our method can detect and segment the currently moving objects given sequential point cloud data exploiting its range projection. Instead of detecting all *potentially movable* objects such as vehicles or humans, our approach distinguishes between *actually moving* objects (labeled in red) and static or non-moving objects (black) in the upper row. At the bottom, we show the range image and our predictions in comparison to the ground truth labels.

static or non-moving objects such as buildings, parked cars, etc. See Figure 5.1 for an example scene and our segmentation. Moving objects are colored in red. We propose a novel approach based on convolutional neural networks (CNNs) to explicitly address the MOS problem for 3D LiDAR scans. We exploit range images as an intermediate representation, which is a natural representation of the scan from a rotating 3D LiDAR such as a Velodyne or Ouster sensors. Based on this comparably light-weight representation, we can directly exploit the existing range-image-based semantic segmentation networks as proposed by Milioto *et al.* (*Milioto et al.*, 2019b), Cortinhal *et al.* (*Cortinhal et al.*, 2020a), and Li *et al.* (*Li et al.*, 2020b) to tackle the MOS problem. Most of such existing LiDAR-based semantic segmentation networks predict the semantic labels of a point cloud, *e.g.*vehicle, building, road, etc. They do not distinguish between actually moving objects, like moving cars, and static objects, like parked cars and also buildings, etc. We are making this distinction and are exploiting sequences of range images, allowing for an effective moving object segmentation targeted to autonomous vehicles. Our main application focus is the perception of self-driving cars in outdoor scenes, but the method itself is not restricted to this domain.

This chapter's main contribution is a novel method based on CNNs using range images generated from 3D LiDAR scans together with the residual images generated from past scans as inputs and outputs for each range measurement in the current frame a label indicating if it belongs to a moving object or not. By combining range images and residual images our network exploits the temporal information and can differentiate between moving and static objects as shown in Figure 5.1. For training, we reorganize the SemanticKITTI (*Behley et al.*, 2019a) dataset and merge the original labels into two classes, moving and static, by exploiting the existing annotations of moving traffic

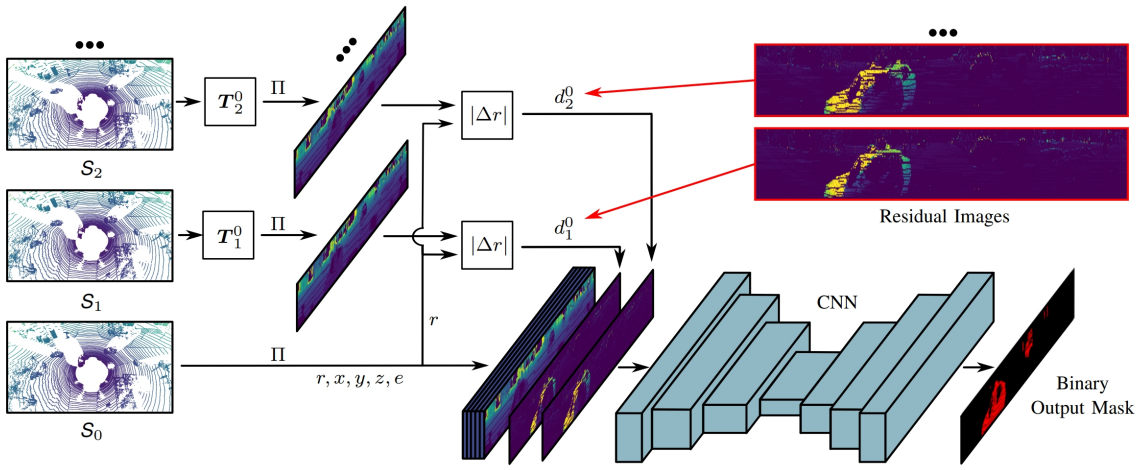


Figure 5.2: Overview of our method. We use the range projection-based representation of LiDAR scans to achieve online moving object segmentation. Given the current scan S_0, we generate residual images from previous scans $\{S_i\}_{i=1}^{N}$ to explore the sequential information. This is by transforming them to the current viewpoint with a homogeneous transformation matrix $\underline{\mathrm{T}}_i^0$ estimated from a SLAM or sensor-based odometry, projecting them to the range representation with a mapping Π and subtracting them from the current scan's range image. The residual images are then concatenated with the current range image and used as input to a fully convolutional neural network. Trained with the binary labels, the proposed method can separate moving and static objects.

participants. Furthermore, our approach runs faster than the sensor frame rate, *i.e.*, 10 Hz for a typical rotating 3D LiDAR sensor. Comparisons with multiple existing methods suggest that the proposed approach leads to more accurate moving object segmentation. In sum, we make two key claims: First, our approach is able to achieve moving object segmentation using only 3D LiDAR scans and runs faster than the sensor frame rate of 10 Hz. Second, it improves the moving object segmentation performance by incorporating residual images in addition to the current scan and outperforms several state-of-the-art networks. To allow for as easy as possible comparisons and support future research, we propose and release a moving object segmentation benchmark, including a hidden test set, based on the SemanticKITTI dataset and will release the source code of our approach.

5.2 Our Approach

The goal of our approach is to achieve accurate and fast moving object segmentation (MOS) for LiDAR scans to enable autonomous mobile systems to make decisions in a timely manner. Figure 5.2 shows a conceptual overview of our proposed method. To achieve online MOS, we first project the point clouds into a range image representation by spherical projection. To separate moving and non-moving objects, we then exploit sequential information (see Section 5.2.1) computing residuals between the current and the previous scans (see Section 5.2.2). We finally concatenate them together with the range information as the input for a segmentation CNN (see Section 5.2.3). In addition, we propose a novel MOS benchmark based on SemanticKITTI (see Section 5.2.4) to train and evaluate MOS methods.

5.2.1 Sequence Information

We aim at segmenting moving objects online, *i.e.*, only using the current and recent LiDAR scans, such that one can exploit the information for odometry estimation in a SLAM pipeline and potentially remove dynamics from a map representation. We assume that we are given a time series of N LiDAR scans in the SLAM history, denoted by $\boldsymbol{S}_j = \{\boldsymbol{p}_i \in \mathbb{R}^4\}$ with M points represented as homogeneous coordinates, *i.e.*, $\boldsymbol{p}_i = (x, y, z, 1)$. We denote the current LiDAR scan by $\boldsymbol{S}_0$ and the sequence of N previously scans by $\boldsymbol{S}_j$ with $1 < j < N$. The *estimated* N consecutive relative transformations from the SLAM / odometry approach, $\boldsymbol{T}_N^{N-1}, \ldots, \boldsymbol{T}_1^0$, between the $N + 1$ scanning poses, represented as transformation matrices, *i.e.*, $\boldsymbol{T}_k^l \in \mathbb{R}^{4\times4}$, are also assumed to be available. Given the estimated relative poses between consecutive scans, we can transform points from one viewpoint to another. We denote the k^{th} scan transformed into the l^{th} scan's coordinate frame by

$$\boldsymbol{S}^{k\to l} = \{\boldsymbol{T}_k^l \boldsymbol{p}_i | \boldsymbol{p}_i \in \boldsymbol{S}_k\}, \tag{5.1}$$

with $\boldsymbol{T}_k^l = \prod_{j=k}^{l+1} \boldsymbol{T}_j^{j-1}$.

5.2.2 Residual Images

Inspired by *Wang et al.* (2018), who exploit the difference between RGB video frames for action recognition, we propose to use LiDAR-based residual images together with pixel-wise binary labels on the range image to segment moving objects. Combining the current sensor reading and residual images, we can employ existing segmentation networks to distinguish between pixels on moving objects and backgrounds by leveraging the temporal information inside the residual images.

To generate the residual images and later fuse them into the current range image, transformation, and re-projection are required. To realize this, we propose a three-step procedure: First, we compensate for the ego-motion by transforming the previous scans into the current local coordinate system given the transformation estimates as defined in Equation 5.1. Next, the transformed past scans $\boldsymbol{S}^{k\to l}$ are re-projected into the current range image view by spherical projection. We compute the residual $d_{k,i}^l$ for each pixel i by computing the normalized absolute difference between the ranges of the current frame and the transformed frame by

$$d_{k,i}^l = \frac{\operatorname{abs} r_i - r_i^{k\to l}}{r_i}, \tag{5.2}$$

where r_i is the range value of $\boldsymbol{p}_i$ from the current frame located at image coordinates (u_i, v_i) and $r_i^{k\to l}$ is the corresponding range value from the transformed scan located at the same image pixel. We only calculate the residual for the valid pixels that contain measurements and set the residual to zero for the invalid pixels. Examples of such residual images are depicted in Figure 5.3. We can see that due to the motion of objects in the scene, *e.g.*the moving car, the displacement between these points in the common viewpoint is relatively large compared to the static background. However, there are ambiguities, since the large residual patterns appear twice for one moving object, while for the slowly moving objects the residual patterns are not obvious. Therefore, directly using residual images for moving object segmentation does not lead to a great performance. It, however, provides a valuable cue for moving objects and can guide the network to separate moving and non-moving objects.

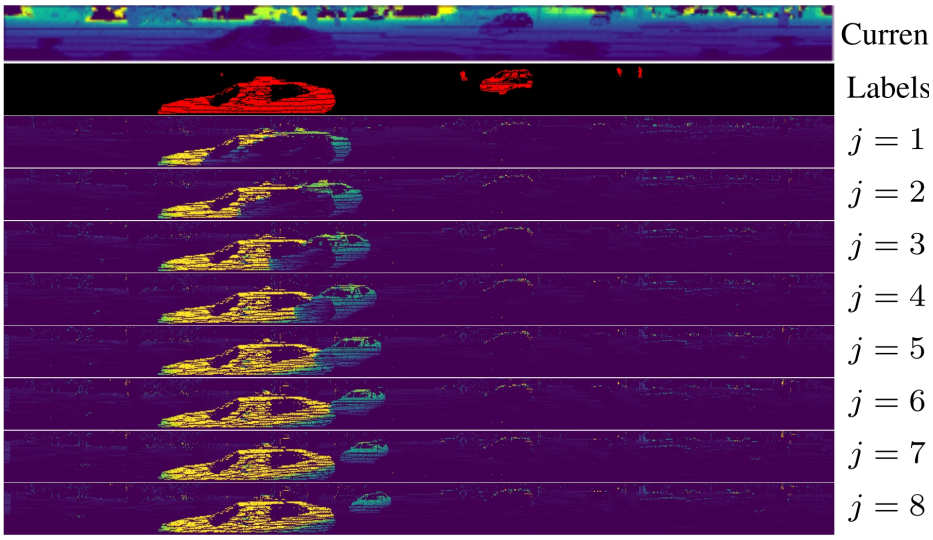


Figure 5.3: Residual images, where j means the residual image generated between the current frame and the last j-th frame. We can see the continuous discrepancy in the residual images due to the motion of the moving car.

In the end, the residual images are concatenated with the current range image as extra channels where range image provides spatial information and residual image encodes temporal information. Each pixel (u_i, v_i) in the fused range image then contains a vector of different types of information $(x_i, y_i, z_i, r_i, e_i, d^0_{1,i}, ..., d^0_{j,i}, ..., d^0_{N-1,i})$, where d^0_j is the residual image calculated between the last j^{th} frame and the current frame.

5.2.3 Range Projection-based Segmentation CNNs

We do not design a new network architecture but reuse networks that have been successfully applied to LiDAR-based semantic segmentation in the past. We adopt and evaluate three popular networks, namely SalsaNext (*Cortinhal et al.*, 2020a), RangeNet++ (*Milioto et al.*, 2019a), and MINet (*Li et al.*, 2020b), for MOS. SalsaNext and RangeNet++ are encoder-decoder architectures with a solid performance and MINet uses a light-weight and efficient multi-path architecture. After the segmentation, a fast GPU-based k-Nearest-Neighbor search over the point cloud is used to remove artifacts produced by the range projection (*Milioto et al.*, 2019a). All methods are state-of-the-art range projection-based LiDAR semantic segmentation networks, comparably light-weight, and can achieve real-time operation, *i.e.*, run faster than the frame rate of the employed LiDAR sensor, which is 10 Hz for common Ouster and Velodyne scanners. For more detailed information about each network, we refer to the original papers (*Milioto et al.*, 2019a; *Cortinhal et al.*, 2020a; *Li et al.*, 2020b).

Instead of changing the architecture of these segmentation networks, we directly feed them with the fused range images plus the residual information, retrain the network and evaluate their performance with our MOS benchmark proposed in Section 5.2.4. Using our proposed residual image approach, all segmentation networks show a large improvement in moving object segmentation as shown in Section 5.3.1. For training, we use the same loss functions as used in the original segmentation methods, while mapping all classes into two per-point classes, moving and non-moving.

5.2.4 Moving Object Segmentation Benchmark

Large datasets for LiDAR-based odometry, object detection, and tracking, like the KITTI Vision Benchmark (*Geiger et al.*, 2012a), and semantic segmentation, panoptic segmentation, and scene completion like SemanticKITTI (*Behley et al.*, 2019a) are available and widely used. There are, however, not many datasets and benchmarks available for 3D LiDAR-based moving object segmentation. With this work, we also aim at covering this gap with a novel benchmark task for MOS.

Our proposed MOS benchmark is based on SemanticKITTI. It uses the same split for training and test set as used in the original odometry dataset, where sequences 00 to 10 are used for training and sequences 11 to 21 are used as a test set. SemanticKITTI contains in total 28 semantic classes such as vehicles, pedestrians, buildings, roads, etc. and distinguishes between moving and non-moving vehicles and humans. In the proposed MOS benchmark, we manually reorganize all the classes into only two types: moving and non-moving/static objects. The actually moving vehicles and humans belong to moving objects and all other classes belong to the non-moving/static objects.

For quantifying the MOS performance, we use the commonly applied Jaccard Index or intersection-over-union (IoU) metric (*Everingham et al.*, 2010) over moving objects, which is given by

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \tag{5.3}$$

where TP, FP, and FN correspond to the number of true positive, false positive, and false negative predictions for the moving class.

5.3 Experimental Evaluation

This chapter focuses on moving object segmentation from 3D LiDAR scan sequences. We present our experiments to show the capabilities of our method and to support our key claims, that our approach: (i) achieve moving object segmentation using only 3D LiDAR scans and runs faster than the sensor frame rate of 10 Hz and (ii) improves the moving object segmentation performance by using residual images, and outperforms several state-of-the-art networks.

We evaluate all the methods on our proposed MOS benchmark, now available online [1]. We use the odometry information provided by SemanticKITTI, which are estimated with a LiDAR-based SLAM system, SuMa (*Behley and Stachniss*, 2018). Aiming at an easy-to-integrate algorithm, we stick to the original setup while only changing the input and the output of the classification head into the proposed binary labels. We train each network using their specific training hyperparameters over 150 epochs on sequences 00-07 and 09-10 and keep sequence 08 as the validation set. For more details on the training regime for each network, we refer to the original papers (*Milioto et al.*, 2019a; *Cortinhal et al.*, 2020a; *Li et al.*, 2020b).

5.3.1 Ablation Study on Input and Architecture

The first ablation study presented in this section is designed to support our claim that our approach is able to achieve moving object segmentation using only 3D LiDAR scans. All the experiments in this section are evaluated on the validation set, *i.e.*, sequence 08.

[1] See http://bit.ly/mos-benchmark for more information.

Input	RangeNet++	MINet	SalsaNext
One frame	38.9	9.1	51.9
Two frames	40.6	35.0	56.0
Residual frames (N=1)	40.9	38.9	59.9

Table 5.1: Evaluating our method with three different networks

We test three different setups with three different networks, RangeNet++, SalsaNext, and MINet, for moving object segmentation as shown in Table 5.1. The first setup is to train the three range projection-based networks directly with the labels for moving and non-moving classes. The second setup is to attach the previous frames to the current frame as the input of the network resulting in 2×5 input channels, as each image contains the coordinates (x, y, z), the range r, and the remission e for each pixel. The third setup is to concatenate the proposed residual images to the current frame as the input of the network and therefore the input size is $5+N$, as detailed in Section 5.2.2.

As can be seen in Table 5.1, RangeNet++ and SalsaNext show a basic performance while MINet fails when training the network together with the binary labels and no additional inputs. Overall, the performance has space for improvement. This is probably due to the fact, that from one frame, the semantic segmentation networks cannot distinguish well between the moving and static objects from the same general class, but may learn some heuristics, *e.g.*that cars on the road are usually moving while those on parking lots are static, which can also be seen in the qualitative results Figure 5.5. A reason why MINet fails may be due to the lightweight architecture that is not capable of learning such heuristics.

In the second setup, we directly combine two frames. Here, the networks can already gain some improvements in MOS, since they can obtain the temporal information from two scans. In this setting, MINet is also capable of predicting moving objects. In the third setup, the best MOS performance is achieved. We hypothesize that it is advantageous to give direct access to the residual information instead of the full range views. Given that most of the points are redundant in two successive frames and the input is large due to the concatenation, the networks need less time to extract the temporal information if the residuals are provided directly. While a large enough network should be able to learn concepts as the difference between frames given enough time, it is generally advantageous to directly provide this information as also shown by *Milioto et al.* (2019a).

As shown in Figure 5.4, we provide two further ablation studies using SalsaNext as the segmentation network. The left figure shows an ablation study on the number of residual images used for MOS Both ablation studies use SalsaNext as the segmentation network. We can see that $N = 1$ residual image attains the biggest improvement in terms of MOS performance while adding more residual images improves the MOS performance further with diminishing returns for $N > 8$ residual images. The figure on the right shows an ablation study on the MOS performance vslet@tokeneonedotthe amount of noise added to the relative odometry poses used to generate the residual images. We manually add noise to the poses estimated by SLAM in (x, y, yaw) with a unit of $(0.1\,m, 0.1\,m, 1^\circ)$ to see how the pose estimations influence our method during inferring. As can be seen, the MOS performance will drop due to the noisy poses. However, when the added noises are larger than 20 units, $(2\,m, 2\,m, 20^\circ)$, the network may ignore the noisy residual images and the MOS performance will not become worse.

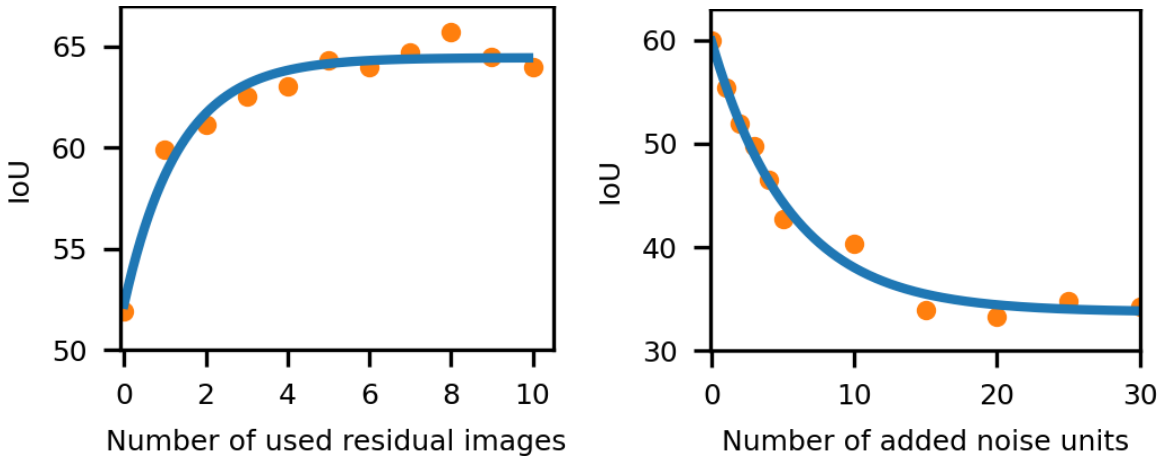


Figure 5.4: Ablation studies. The left figure shows the ablation study on the MOS performance vslet@tokeneonedotthe number of residual images N. The right figure shows the ablation study on the MOS performance vslet@tokeneonedotthe number of added noise units to the poses during the inferring.

5.3.2 MOS Performance and Comparisons

The experiment presented in this section investigates the MOS performance of our approach. It supports the claim that our approach improves the MOS performance by using residual images and outperforms several state-of-the-art networks. Since there are not many existing implementations for LiDAR-based MOS available, we choose several methods that have been used in similar tasks, *e.g.*, semantic segmentation and scene flow, and modify them to achieve LiDAR-based MOS. All the methods are evaluated on the test data of the proposed benchmark, *i.e.*, sequences 11-21.

We analyze multiple alternative approaches. We start using an existing semantic segmentation network, *e.g.*SalsaNext (*Cortinhal et al.*, 2020a), directly and label all the movable objects, *e.g.*vehicles and humans, as moving objects while labeling other objects as static. We name this method as *SalsaNext (movable classes)*. Here, we also show the results generated by the retrained SalsaNext with the proposed binary labels, named *SalsaNext (retrained)*. Since the residual images can already point out rough positions of moving objects, here we also take it as a simple heuristic-based baseline, named *Residual*. Inspired by Yoon *et al.* (*Yoon et al.*, 2019), we also re-implement the pure geometric heuristic-based method using residual information together with free space checking and region growing, named as *Residual+RG*.

We furthermore compare our method to the state-of-the-art scene flow method, FlowNet3D (*Liu et al.*, 2019a), referred to as *SceneFlow*, which is a network estimating the translational flow vector for every LiDAR point given two consecutive scans as input. We set a threshold on the estimated translation of each point to decide the label for each point, *i.e.*, points with translations larger than the threshold are labeled as moving. We fix the threshold based on the best MOS performance on the validation set. We also compare our method to the state-of-the-art multiple point cloud-based semantic segmentation methods (*Thomas et al.*, 2019a; *Shi et al.*, 2020a), since they can also distinguish between moving and non-moving classes.

For the non-semantic-based methods, we additionally add semantic information by checking if

	IoU
SalsaNext (moveable classes)	4.4
SalsaNext (retrained)	46.6
Residual	1.9
Residual + RG	14.1
Residual + RG + Semantics	20.6
SceneFlow	4.8
SceneFlow + Semantics	28.7
SqSequence	43.2
KPConv	60.9
Ours (based on SalsaNext/N = 1)	52.0
Ours (based on SalsaNext/N = 8 + Semantics)	**62.5**

Table 5.2: MOS performance compared to the state of the art.

the predicted moving objects are movable or not, and only label a point as moving if it is both predicted as moving by the original method and at the same time assigned to a movable object, *e.g.*vehicles and humans. The semantic information is generated using SalsaNext with the pre-trained weights provided by the original paper. We identify the semantic-enhanced methods by adding *+Semantics*.

We compare two setups of our method to all the above-mentioned methods. For our methods, we choose SalsaNext as the base network as it shows the best performance in our ablation study. In the first setup, we use only one residual image, $N = 1$, to obtain the temporal information, and in the other setup, we use our best setup fixed on the validation sequence with $N = 8$ residual images and semantic information to see the best performance of our method.

As shown in Table 5.2, our residual image-based method with $N = 1$ already outperforms most baselines, while being worse than KPConv, which is a dense multiple point clouds-based semantic segmentation method. Due to the heavy computation burden, it cannot achieve real-time performance. When our method uses multiple residual images ($N = 8$) together with semantic information, our method outperforms all other methods.

Figure 5.5 and Figure 5.6 show the qualitative results on range images and LiDAR scans respectively in a very challenging situation, where the car is at the intersection and there are both a lot of moving objects and static objects. Our method can distinguish moving and static points even when some of the moving objects are moving slowly and other methods fail to detect this.

5.3.3 Applications

Two obvious applications of our proposed method are LiDAR-based odometry/SLAM as well as 3D mapping. Here, we show the effectiveness of our method by using the MOS predictions as masks for the input LiDAR scans, which removes effectively all points belonging to moving objects. No further tweaks have been employed. We use our best setup for the MOS, *i.e.*, our approach extending

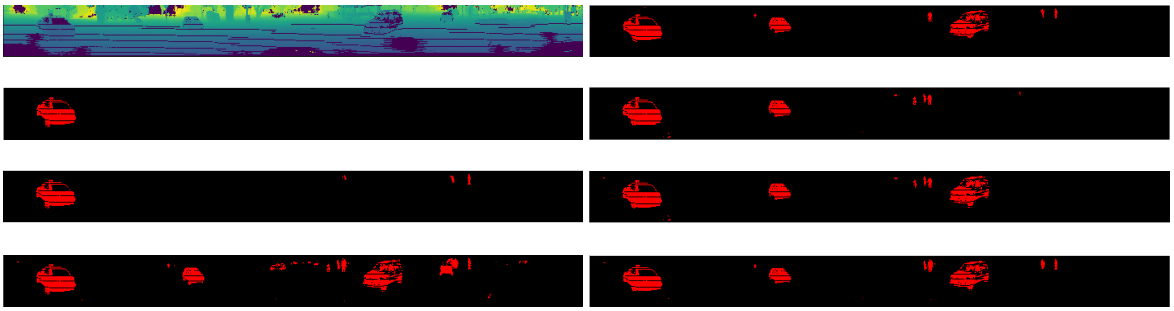


Figure 5.5: Qualitative results with range projections, where red pixels correspond to moving objects.

Split	Approach		
	SuMa	SuMa++	SuMa+MOS
Train (Seq. 00-10)	0.36/0.83	0.29/0.70	**0.29/0.66**
Test (Seq. 11-21)	0.34/1.39	0.34/1.06	**0.33/0.99**

Table 5.3: KITTI Odometry Benchmark Results. Relative errors averaged over trajectories of 100 to 800 m length: relative rotational error in degrees per 100 m / relative translational error in %.

SalsaNext with $N = 8$ residual images and semantics.

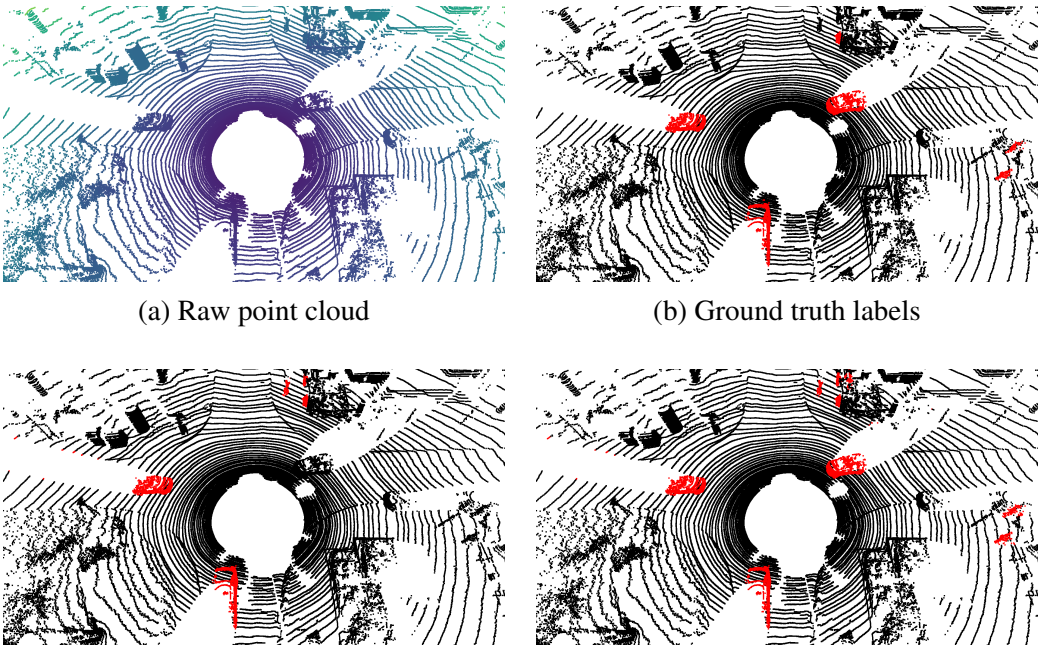
5.3.3.1 Odometry/SLAM

For the LiDAR-based odometry experiments, we use an off-the-shelf SLAM approach (*Behley and Stachniss*, 2018) and apply our MOS method before feeding the point cloud into the SLAM pipeline. We compare the improved odometry results to both the original approach, called SuMa, and our semantic-enhanced approach, SuMa++ (*Chen et al.*, 2019a). We evaluate these odometry methods, SuMa, SuMa++, and SuMa+MOS on the KITTI odometry benchmark (*Geiger et al.*, 2012a).

The quantitative results are shown in Table 5.3. We can see that, by simply applying our MOS predictions as a pre-processing mask, the odometry results are improved in both the KITTI training and test data and even slightly better than the well-designed semantic-enhanced SuMa.

5.3.3.2 3D Mapping

As shown in Figure 5.7, we compare the aggregated point cloud maps (a) directly with the raw LiDAR scans, (b) with the cleaned LiDAR scans by applying our MOS predictions as masks. We use the Open3D library (*Zhou et al.*, 2018) to visualize the mapping results. As can be seen, there are moving objects present that pollute the map, which might have adversarial effects, when used for localization or path planning. By using our MOS predictions as masks, we can effectively remove these artifacts and get a clean map. Note that, here we show two direct use cases of our MOS approach without any further optimizations employed.



(a) Raw point cloud (b) Ground truth labels

(c) SalsaNext (retrained) (d) SalsaNext+ N = 8 + Semantics (Ours)

Figure 5.6: Qualitative results shown as point clouds. (a) shows the raw point cloud with points colored depending on the range from purple (near) to yellow (far). (b) shows the ground truth, and (c,d) prediction results, where red points correspond to the class moving.

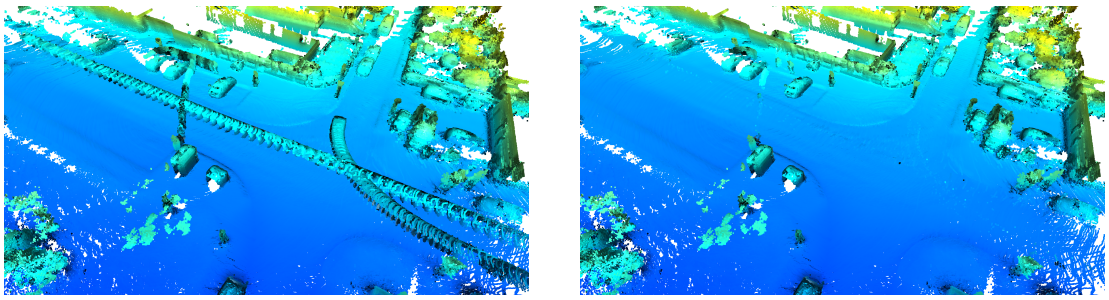
5.3.4 Runtime

The runtime is evaluated on sequence 08 with an Intel i7-8700 with 3.2 GHz and a single Nvidia Quadro P6000 graphic card. It takes around 10 ms on average to estimate the odometry and generate the residual image. Since we only change the input of each network while keeping the architecture the same, the inference time is nearly the same as before, specifically 75 ms for RangeNet++, 42 ms for SalsaNext, and 21 ms for MINet. In case of using semantics for the MOS, we can run a second full semantics network in parallel.

As the odometry history for the SLAM is available, we need to estimate the pose and generate the residual images only once for every incoming frame. In sum, using our method for LiDAR-based odometry takes approx. 51 ms per scan (=20 Hz) using SalsaNext, which is faster than the frame rate of a typical LiDAR sensor, *i.e.*, 10 Hz.

5.4 Summary

In this chapter, we presented a novel and effective approach to achieve LiDAR-based moving object segmentation in an end-to-end online fashion. Our method exploits neural networks and sequential information, which allows our approach to successfully distinguish between moving and static objects. Our method is based on range projections and thus fast and can directly improve existing SLAM and mapping systems. The experiments suggest that our method achieves good performance on MOS and outperformed several state-of-the-art approaches. We also propose a new benchmark



(a) Raw point clouds (b) Point clouds with moving segments removed

Figure 5.7: Mapping Results on Sequence 08, Frame 3960-4070, where we show the accumulated point cloud (a) without removing segments and (b) when we remove the segments predicted as moving.

for LiDAR-based MOS and used it to evaluate our approach, also allowing further comparisons with future MOS systems.

CHAPTER 6

Rethinking 3D LiDAR Point Cloud Segmentation

Individual Contribution

The following chapter is based on the publication:

Rethinking 3D LiDAR Point Cloud Segmentation
ShiJie Li, Yun Liu, Juergen Gall
IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2021.
doi:10.1109/TNNLS.2021.3132836

Below we will summarize the contributions of each author.

- **Shijie Li**
  Shijie proposed the main idea of the paper and implemented the whole method. Shijie performed the experiments and conducted the qualitative and quantitative analysis. Shijie also wrote most of the paper.
- **Yun Liu**
  Yun contributed with writing.
- **Juergen Gall**
  Juergen supervised the project. He contributed with discussion and writing.

In the previous chapters, we have proposed a very efficient projection-based architecture for both LiDAR point cloud semantic segmentation and moving object segmentation. Apart from projection-based methods, point-based methods, which directly operate on the raw point clouds, are also widely used in 3D scene understanding. However, most of them have been designed for indoor scenarios, but they struggle if they are applied to point clouds that are captured by a LiDAR sensor in an outdoor environment.

In this chapter, itomake these methods more efficient and robust such that they can handle LiDAR data, we introduce the general concept of reformulating 3D point-based operations such that they can operate in the projection space. While we show using three point-based methods that the reformulated versions are between 300 and 400 times faster and achieve a higher accuracy, we furthermore demonstrate that the concept of reformulating 3D point-based operations allows to design of new architectures that unify the benefits of point-based and image-based methods. As an example, we introduce a network that integrates reformulated 3D point-based operations into a 2D encoder-decoder architecture that fuses the information from different 2D scales. We evaluate the approach

on four challenging datasets for semantic LiDAR point cloud segmentation and show that leveraging reformulated 3D point-based operations with 2D image-based operations achieves very good results for all four datasets.

Contents

6.1 Introduction

In this work, we address these issues and demonstrate that point-based methods can be reformulated such that they are suitable for LiDAR data. The core idea is that we make use of a projection of the LiDAR point cloud as shown in the bottom row of Figure 6.1. In contrast to methods (*Wu et al.*, 2018b, 2019b; *Milioto et al.*, 2019b) that apply 2D convolutions, we preserve the architectures and the operations of point-based methods. Point-based methods comprise several steps that are repeated within the network architecture. These steps include the sampling of 3D points of the point cloud, grouping neighboring points for each sampled point, and computing a feature based on the grouped points. In this work, we show how these operations can be performed in the projection space and how these operations can be efficiently implemented. Although the operations are the same, the projection leads to significant differences. For instance, the sampled points are differently distributed as shown in Figure 6.3. The sampled points of the projected-point version are actually better distributed than the sampled points of the point-based methods that oversample the sparse distant points in a LiDAR point cloud.

We demonstrate the general concept of reformulating point-based methods by means of the three point-based methods PointNet++ (*Qi et al.*, 2017d), SpiderCNN (*Xu et al.*, 2018), and PointConv (*Wu et al.*, 2019c) and show that the reformulated versions are between 300 and 400 times faster and increase the mIoU by 58% - 68%. While the reformulated versions preserve the operations of the original point-based methods, we also demonstrate that the concept of reformulating point-based methods can also be used to develop new architectures that leverage reformulated 3D point-based operations with 2D image-based operations. As an example of such a network, we propose a network for 3D LiDAR point cloud segmentation, which we term Unprojection Network (UnPNet). It integrates the reformulated feature propagation of PointConv for up- and down-sampling into a 2D

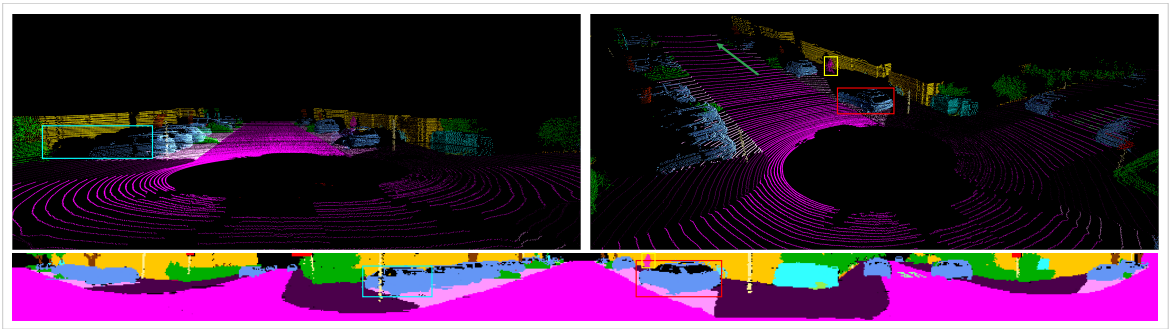


Figure 6.1: Due to the nature of LiDAR sensors, the captured 3D point cloud can be projected onto a plane. This means that neighboring points in 3D are also close in the projected plane, but neighbors in the projected plane can be far distant in 3D. Furthermore, the points become more sparse as the distance to the sensor increases (green arrow), while the points are dense in the projection. We highlight some objects by bounding boxes as an example. Best seen using the zoom function of a PDF viewer.

encoder-decoder architecture. In this way, we exploit 3D operations that are reformulated to operate in the projection space as well as 2D operations that fuse the information from different 2D scales. Furthermore, we employ edge supervision which would be impossible for point-based methods.

We evaluate UnPNet and the reformulated versions of PointNet++ (*Qi et al.*, 2017d), SpiderCNN (*Xu et al.*, 2018), and PointConv (*Wu et al.*, 2019c) on the SemanticKITTI dataset (*Behley et al.*, 2019d), which is a large-scale dataset for semantic segmentation of LiDAR point clouds. Apart from SemanticKITTI, we also evaluate the proposed UnPNet on three other datasets for a comprehensive comparison. The experiments show that UnPNet performs very well on all four datasets and that it outperforms the reformulated point-based methods since it combines 3D point operations with 2D fusion techniques.

In summary, we show in this work that

- point-based methods can be reformulated such that they operate in the projection space for processing LiDAR point clouds;
- the reformulated point-based methods are more efficient and achieve a higher accuracy than the original point-based methods;
- The combination of reformulated 3D point-based operations with 2D image-based operations unifies the benefits of point-based and image-based methods.

Code will be released at `https://github.com/sj-li/UnpNet`.

6.2 Reformulation of Point-based methods

In order to show how a network operating on LiDAR points can be reformulated to operate in the projection space, we use PointNet++ (*Qi et al.*, 2017d) as an example. In Section 6.2.3, we discuss the reformulated examples of two other point-based networks, namely SpiderCNN (*Xu et al.*, 2018) and PointConv (*Wu et al.*, 2019c). Before we discuss our approach in Section. 6.2.2, we briefly discuss the main operations of PointNet++.

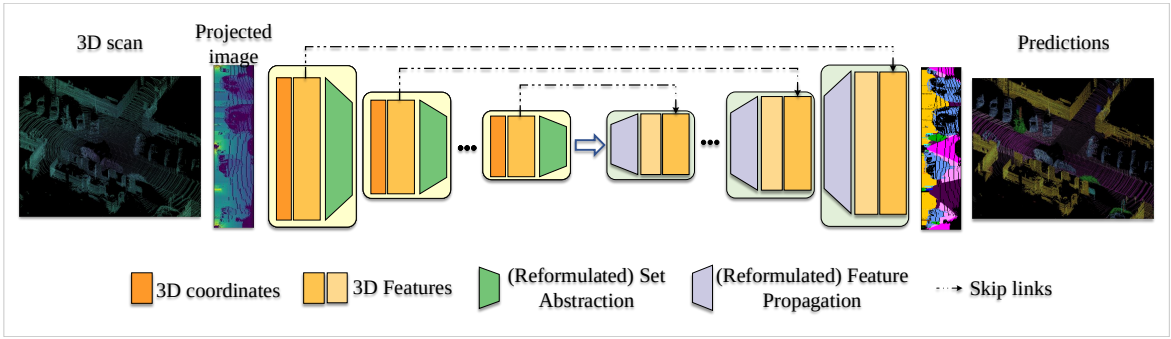


Figure 6.2: Overall architecture of PointNet++ or its reformulation (reformulated PointNet++).

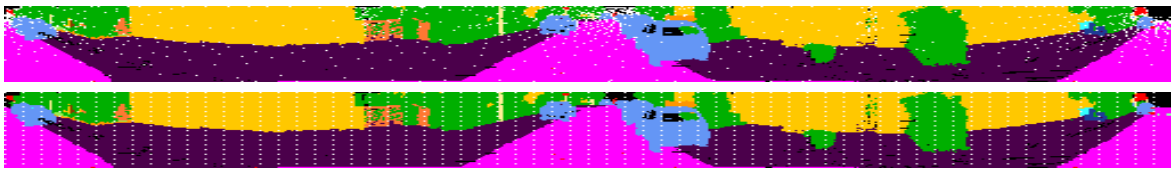


Figure 6.3: Comparison of the sampling methods in PointNet++ (top) and reformulated PointNet++ (bottom). 1024 points are sampled from the LiDAR point cloud. For better visualization, we show the 2D projected image with white points denoting the sampled points. PointNet++ only samples a few points for close objects like cars and most sampled points lie on the distant region where the real distribution of points is sparse.

6.2.1 Review of PointNet++

We choose PointNet++ (*Qi et al.*, 2017d) as an example since it is very popular and has been used as the baseline in many works. The pipeline of PointNet++ is shown in Figure 6.2. PointNet++ consists of so-called set abstraction modules and feature propagation modules as shown in Figure 6.2. The set abstraction module comprises a sampling layer, a grouping layer, and a PointNet layer. The sampling layer chooses a subset from the input point set, which defines the centroids of local regions. Figure 6.3 shows the sampled points. The grouping layer groups the neighboring points of each centroid, which forms a local region. The PointNet layer computes a feature vector based on the neighboring points using a multilayer perceptron (MLP) and max pooling. While the set abstraction modules subsample the original point set, the feature propagation modules recover the original point set by distance based interpolation:

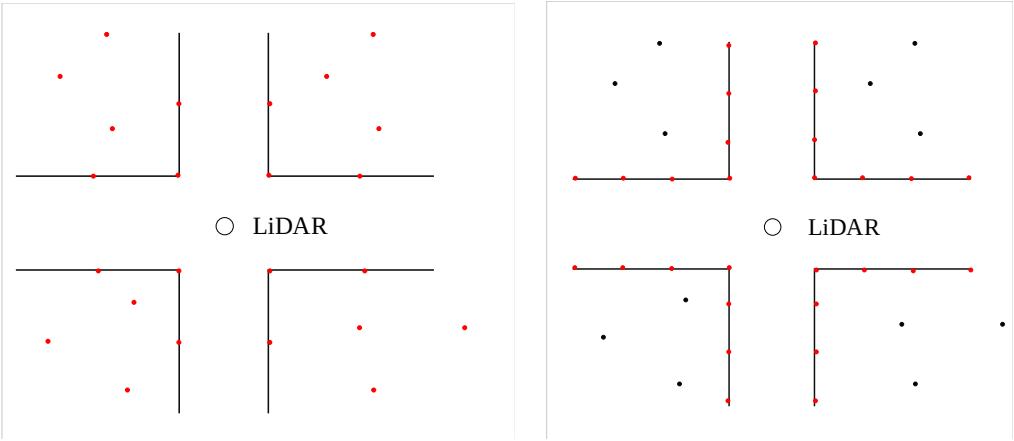
$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)}, \tag{6.1}$$

$$w_i(x) = \frac{1}{d\,(x, x_i)^p}, j = 1, \ldots, C, \tag{6.2}$$

where f is a point-wise feature and $d(x_i, x_j)$ is the distance between point x_i and x_j.

6.2.2 Reformulated PointNet++

To reformulate PointNet++ so that it operates in the projection space, we will not change the architecture, but we need to reformulate the set abstraction and the feature propagation module as shown



(a) Farthest point sampling (b)Ray sampling

Figure 6.4: Toy example. The points on the black lines are correct measurements while the other points are outliers. Farthest point sampling selects the outliers such that the sampled points are uniformly distributed in the 3D space. The proposed ray sampling only selects the points (red) on the black lines. The outliers (blue) are not selected.

in Figure 6.2. As input, we use the projected LiDAR point cloud as it has been proposed in (*Wu et al.*, 2018b).

We will first describe the sampling layer (Section. 6.2.2.1), the grouping layer (Section. 6.2.2.2), and the PointNet layer (Section. 6.2.2.3) of the reformulated set abstraction and then discuss the reformulated feature propagation (Section. 6.2.2.4).

6.2.2.1 Reformulated Sampling Layer

PointNet++ (*Qi et al.*, 2017d) uses farthest point sampling to sample a subset of 3D points. It is designed to maximize the distance between sampled points that are thus uniformly scattered in the 3D space. However, the real distribution of LiDAR points is not uniform and becomes sparse as the distance to the sensor increases as shown in Figure 6.1. This mismatch harms the performance when applying farthest point sampling to LiDAR points, as shown in Figure 6.3. Furthermore, the computational complexity of farthest point sampling is $\mathcal{O}(N \log N)$ where N is the number of 3D points (*Kamousi et al.*, 2016). This makes the approach highly inefficient for large point clouds which are common for LiDAR sensors. Therefore, farthest point sampling is suboptimal for LiDAR point cloud segmentation in terms of both effectiveness and efficiency. To address this problem, we propose to uniformly sample the 3D points from the projected point cloud as shown in Figure 6.3. This has the advantage that we sample rays instead of points, which means that the distance between the sampled points is larger if they are farther away from the sensor. Hence, the distribution of sampled points accords with the original point cloud. The sampling is also less sensitive to outliers. Since farthest point sampling aims to sample points that are uniformly in the 3D space, it tends to select all outliers that are distant to correct measurements. In case of ray sampling, the probability to select an outlier is equivalent to the percentage of outliers and thus lower compared to farthest point sampling as it is illustrated in Figure 6.4. Another benefit is that we can use a 2D grid structure for sampling such that the computational complexity becomes $\mathcal{O}(M)$ where $M = H' \times W'$ is the

number of sampled points and $M \ll N$.

6.2.2.2 Reformulated Grouping Layer

For grouping neighboring 3D points, PointNet++ uses a ball query to obtain for each sampled point all points that are within a given distance. While a naive implementation has the complexity of $\mathcal{O}(MN)$, more efficient implementations reduce it using data structures like k-d trees or octrees (*Behley et al.*, 2015). This, however, increases the memory requirements. In order to make the grouping of PointNet++ (*Qi et al.*, 2017d) much more efficient, we search neighboring rays first and then exclude the points that are too far away from the sampled point. We obtain the neighboring rays by taking the $k \times k$ neighbors in the projected point cloud as shown in Figure 6.5. The parameter k provides a trade-off between accuracy and runtime as we will show in the experiments. For each of the k^2 points, we obtain the 3D points and subtract the 3D position of the sampled point to convert the points from global coordinates to local coordinates as in PointNet++. We then compute the norm of each point, *i.e.*, the 3D distance to the sampled point, and mask all points that are within a given distance. The complexity of this operation is $\mathcal{O}(Mk^2)$ where $k^2 \ll \log N$. A comparison between this grouping strategy and the grouping in PointNet++ is displayed in Figure 6.5.

Figure 6.6 illustrates how reformulated sampling and grouping are efficiently implemented in a network. Given the input $\mathbb{R}^{(C+3)\times H\times W}$, where C is the number of feature channels which are concatenated with the 3D coordinates ($C+3$), the unfold operation uniformly samples $H'\times W'$ points as discussed in Section. 6.2.2.1 and copies the corresponding $k \times k$ neighborhood for each sampled point $m \in H' \times W'$. This yields the tensor $F_{in} \in \mathbb{R}^{(C+3)\times k^2\times H'\times W'}$. For each sampled point, we then subtract the 3D coordinate of the sampled point from the coordinates of the corresponding k^2 neighboring points. We finally compute the distance map $\mathbb{R}^{k^2\times H'\times W'}$ and the binary neighborhood mask $\{0,1\}^{k^2\times H'\times W'}$, which is 1 if a point is within the radius of a sampled point. The neighborhood mask defines the grouping for each sampled point.

At this step, we directly compute the inverse distance map, which will be used for the reformulated feature propagation and will be described in the next section, and the inverse density map as described in (*Wu et al.*, 2019c). The latter will be needed for converting PointConv (*Wu et al.*, 2019c) into a reformulated point-based method.

6.2.2.3 Reformulated PointNet Layer

The PointNet layer in PointNet++ uses max pooling and an MLP. Given $F_{in} \in \mathbb{R}^{(C+3)\times k^2\times H'\times W'}$ after the unfold operation and the neighborhood mask, the reformulated PointNet layer can thus be denoted as

$$F_{out} = \text{Pooling}(\text{MLP}(F_{in} \otimes \mathcal{M})). \tag{6.3}$$

The symbol $\otimes$ denotes element-wise multiplication and $\mathcal{M}$ is the neighborhood mask, which has been duplicated $(C+3)$-times to have the same size as F_{in}. Figure 6.9(a) illustrates this operation.

6.2.2.4 Reformulated Feature Propagation Module

As described in Section. 6.2.1 and illustrated in Figure 6.2, the feature propagation modules recover the original point set by the distance based interpolation (Equation 6.1). Since our sampled points are uniformly distributed on the projected image, this can be very efficiently implemented. The

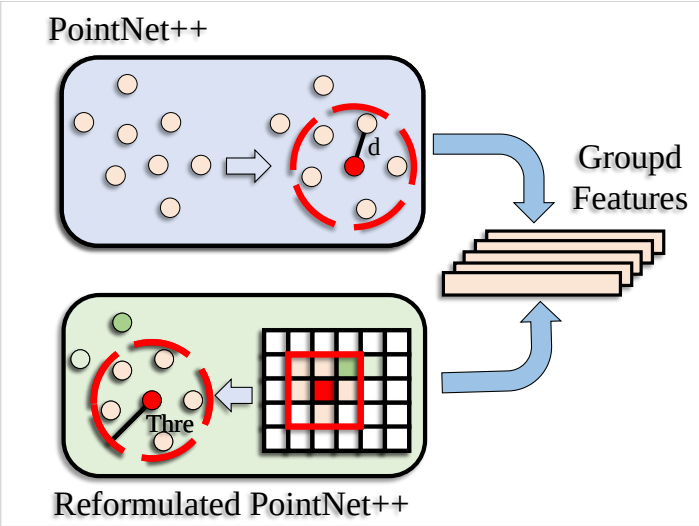


Figure 6.5: Comparison of the reformulated grouping layer (bottom) with the original one in PointNet++ (*Qi et al.*, 2017d) (top). PointNet++ uses a ball query to obtain all neighboring points (orange) within a certain radius for each sampled point (red). In contrast, our method first searches the $k \times k$ neighboring rays, and then we discard the points that are outside the radius (dark green point).

sampled points are first set back to its original positions. The distance based interpolation (Interp) is then applied as in Equation 6.1 using the precomputed inverse distance map $\hat{\mathcal{D}}$. As illustrated in Figure 6.2, there are skip connections between the blocks. The interpolated features $\text{Interp}(F_{in}, \hat{\mathcal{D}})$ are thus concatenated with the point features from the corresponding set abstraction module ($\hat{F}$) and fed into an MLP, *i.e.*,

$$F_{out} = \text{MLP}(\text{Concat}(\text{Interp}(F_{in}, \hat{\mathcal{D}}), \hat{F})). \tag{6.4}$$

The reformulation of the Feature Propagation module is illustrated in Figure 6.7.

6.2.3 Reformulated SpiderCNN and PointConv

So far we discussed how PointNet++ (*Qi et al.*, 2017d) can be reformulated so that it operates in the projection space, but the approach can be applied to other point-based networks as well. In this section, we therefore briefly describe how two other networks, namely SpiderCNN (*Xu et al.*, 2018) and PointConv (*Wu et al.*, 2019c), are reformulated. Figure 6.9 illustrates the differences between the reformulation of PointNet++, SpiderCNN, and PointConv.

The reformulated SpiderCNN (RSpiderCNN) can be viewed as a 'soft' version of RPointNet++ because it weights each point feature according to its relative position to the corresponding sampled point. Instead of using a neighborhood mask, it computes the weight for each point. In our case, the operations are performed for the $k \times k$ neighborhood. The operations of the set abstraction in RSpiderCNN are thus defined by

$$F_{out} = \text{MLP}_{out}(\text{MLP}_{in}(F_{in}) \otimes \textit{WeightNet}(\mathcal{P})), \tag{6.5}$$

where the symbol $\otimes$ denotes matrix multiplication, and $\mathcal{P}$ denotes the unfolded 3D coordinates after subtracting the 3D coordinates of the corresponding sampled point as shown in Figure 6.6. Here, we

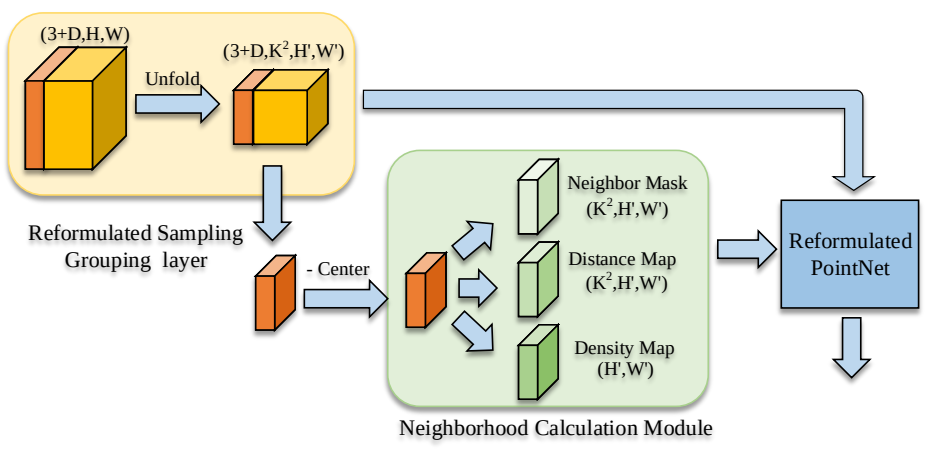


Figure 6.6: Illustration of the reformulated set abstraction.

omit the dimensions which are shown in Figure 6.9(c). For feature propagation, RSpiderCNN follows Equation 6.4 and the main difference is that RSpiderCNN applies Equation 6.5 on the interpolated features.

The reformulated PointConv (RPointConv) uses the point density as additional information. The green branch in Figure 6.9(d) therefore takes the inverse density map $\mathcal{D}$ from Figure 6.6 as input. The operations of the set abstraction in RPointConv are defined by

$$\begin{aligned} F_{out} =& \text{MLP}_{out}((\text{MLP}_{in}(F_{in}) \otimes \\ & \textit{DensityNet}(\mathcal{D})) \otimes \textit{WeightNet}(\mathcal{P})). \end{aligned} \tag{6.6}$$

Similar to RSpiderCNN, RPointConv follows Equation 6.4 for feature propagation, but it applies Equation 6.6 on the interpolated features.

These examples show that point-based methods can be reformulated to operate in the projection space. In the experiments, we will show that the reformulation makes the point-based methods 300-400 times faster and increases the accuracy.

6.3 Unprojection Network (UnPNet)

So far, we have shown how point-based methods can be reformulated without changing the architecture design and principles. The reformulated approaches have the potential to leverage concepts from 3D point-based methods and 2D image-based methods. In order to demonstrate this, we propose a network that uses the reformulated feature propagation of PointConv (Equation 6.6) for up- and down-sampling and we integrate it into a 2D CNN with an encoder-decoder architecture. In this

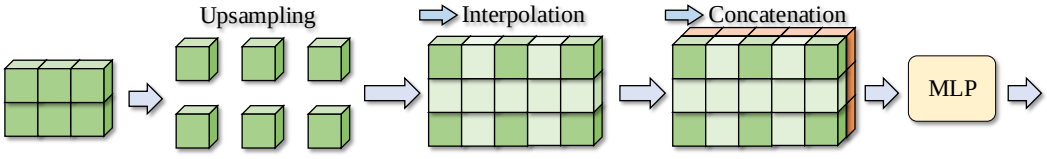


Figure 6.7: Illustration of the reformulated feature propagation.

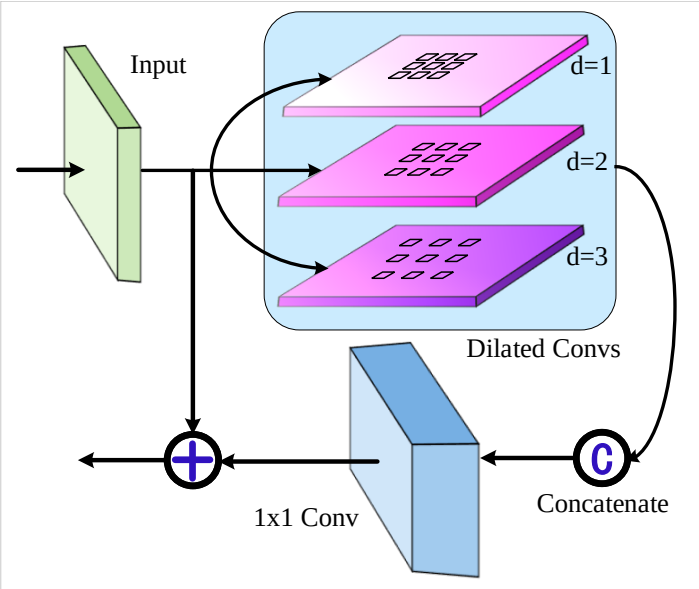


Figure 6.8: Illustration of the context block.

k	Acc	mIoU	Scans/s
3	71.4	26.8	38.2
5	74.7	30.7	30.0
7	76.2	31.9	23.7

((a)) Search size k

R	Acc	mIoU	Scans/s
64×512	74.7	30.7	30.0
64×1024	77.4	31.3	17.2
64×2048	75.2	25.7	9.4

((b)) Input resolution R

S	Acc	mIoU	Scans/s
1	74.5	30.4	22.2
2	74.7	30.7	30.0
4	69.6	25.0	34.5

((c)) Sampling stride S

Table 6.1: Effect of different parameters.

way, we exploit 3D operations that are reformulated to operate in the projection space as well as 2D operations that fuse the information from different 2D scales.

The network architecture is shown in Section. 6.2.2. As in the reformulated architectures, we first project the LiDAR point cloud by spherical projection. Besides the basic block (*i.e.*, 2D convolutions with the residual link), the network uses the reformulated feature propagation of PointConv (Equation 6.6) for up- and down-sampling and an additional context block shown in Figure 6.8 for fusing image context at multiple 2D scales. In order to make the context block as efficient as possible, we use 2D dilated convolutions. More in detail, we use three 3×3 convolutions with different dilation rates (1, 2, 3) to extract multi-scale features. The three feature maps are then concatenated and fused by a 1×1 convolution. In addition, a residual link is employed to facilitate the gradient flow. Furthermore, we apply edge supervision to the decoder, which will be described in Section. 6.3.1, to ensure better segment boundaries. As in *Milioto et al.* (2019b), k-NN can be used for post-processing. We call this architecture Unprojection Network (UnPNet), and we will show in the experiments that UnPNet outperforms RPointConv by a large margin. While UnPNet is just an example, it demonstrates that the reformulation of point-based methods is a new general concept, allowing to construction of new architectures by leveraging 3D point-based networks and 2D CNNs.

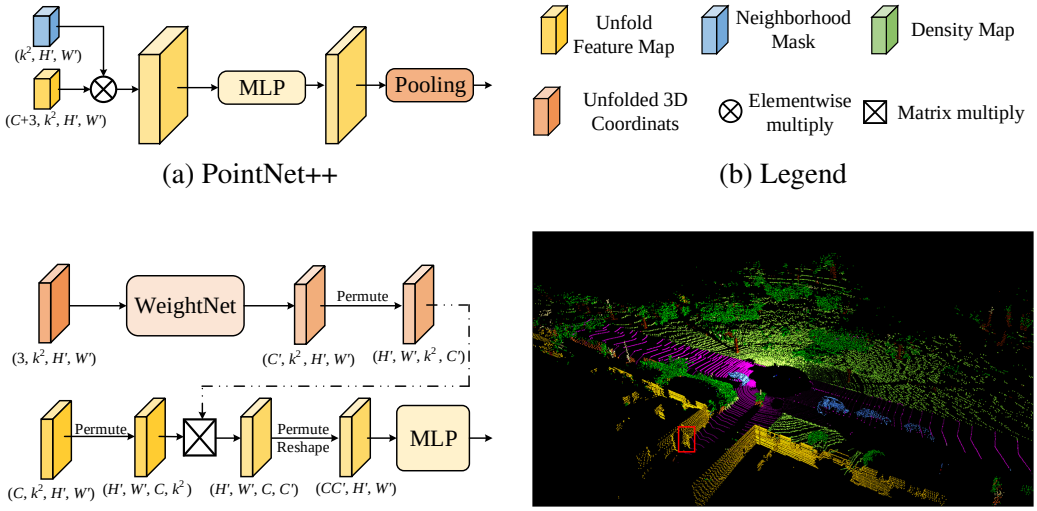


Figure 6.9: Reformulation of the set abstraction module for PointNet++ (*Qi et al.*, 2017d), SpiderCNN (*Xu et al.*, 2018), and PointConv (*Wu et al.*, 2019c). In case of a dimension mismatch in the element-wise operations, we use the broadcasting mechanism, which is omitted in the illustrations since it is the default operation in modern deep learning frameworks like PyTorch (*Paszke et al.*, 2019). We also omit MLP_{in} from Equation 6.5 and Equation 6.6.

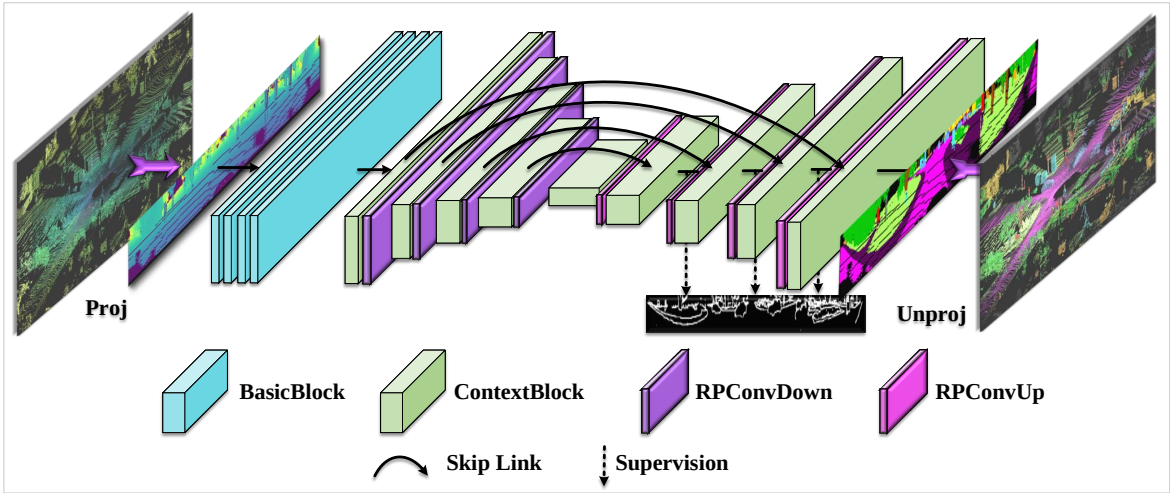


Figure 6.10: Overview of UnPNet. The network combines 2D operations like the basic block and the context block as well as reformulated 3D operations. In this example, we use the reformulated feature propagation of PointConv for down- and upsampling. The corresponding operations are denoted by RPConvDown and RPConvUp, respectively.

	PointNet++ (*Qi et al.*, 2017d)	RPointNet++	SCNN (*Xu et al.*, 2018)	RSCNN	PointConv (*Wu et al.*, 2019c)	RPointConv
Acc	64.6	74.7	70.3	81.5	72.5	82.5
mIoU	19.4	30.7	21.8	36.8	23.2	37.6
Scans/sec	0.1	30.0	0.04	12.9	0.03	12.2

Table 6.2: Comparison with the original baseline methods.

	Acc	mIoU	Scans/s
RPointConv	82.5	37.6	12.2
UnPNet *w/o* supp	87.0	50.2	12.8
UnPNet	87.4	50.7	12.8
UnPNet + *k*-NN	89.1	54.6	12.5

Table 6.3: Ablation study for UnPNet.

6.3.1 Auxiliary Supervision

For the decoder of UnPNet, we employ edge supervision to obtain accurate boundaries of the segments. Before each upsampling operation, we apply edge supervision by computing the edge loss $\mathcal{L}_e$ using the binary entropy loss as

$$\mathcal{L}_e = -\frac{1}{|I|}\sum_{i\in\mathbf{I}}(e_i \log(\hat{e}_i) + (1-e_i)\log(1-\hat{e}_i)), \tag{6.7}$$

where $e_i \in \{0,1\}$ is the ground-truth edge for pixel i and $\hat{e}_i$ is the corresponding predicted probability estimated by a 1×1 convolution. The ground-truth edge is obtained by the segment boundaries of the ground-truth segmentation.

As for the final semantic prediction of UnPNet, we use two loss terms. The first one is the standard weighted cross-entropy loss which can be formulated as

$$\mathcal{L}_s = -\frac{1}{|I|}\sum_{i\in I}\sum_{n=1}^{N} w_n p_i^n \log(\hat{p}_i^n), \tag{6.8}$$

where N is the number of classes, $p_i^n \in \{0,1\}$ is 1 if pixel i is annotated by class n, and $\hat{p}_i^n$ is the predicted class probability. The weight w_n for class n is inversely proportional to its frequency of occurrence.

Apart from the weighted cross-entropy loss, we also directly maximize the intersection-over-union (IoU) score by the Lovász-Softmax loss (*Berman et al.*, 2018a):

$$\mathcal{L}_{ls} = \frac{1}{N}\sum_{n=1}^{N}\overline{\Delta_{J_n}}(m(n)), \tag{6.9}$$

$$m_i(n) = \begin{cases} 1-\hat{p}_i^n & \text{if } p_i^n = 1 \\ \hat{p}_i^n & \text{otherwise,} \end{cases} \tag{6.10}$$

where $\overline{\Delta_{J_n}}$ is the Lovász extension of the Jaccard index.

Hence, the total loss is given by

$$\mathcal{L} = \mathcal{L}_s + \mathcal{L}_{ls} + \frac{1}{2}\sum_{u}\mathcal{L}_e^u, \tag{6.11}$$

where $\mathcal{L}_s$ denotes the weighted cross-entropy loss Equation 6.8, $\mathcal{L}_{ls}$ denotes the Lovász-Softmax loss Equation 6.9, and $\mathcal{L}_e^u$ is the edge loss Equation 6.7 for each upsampling step u of the decoder.

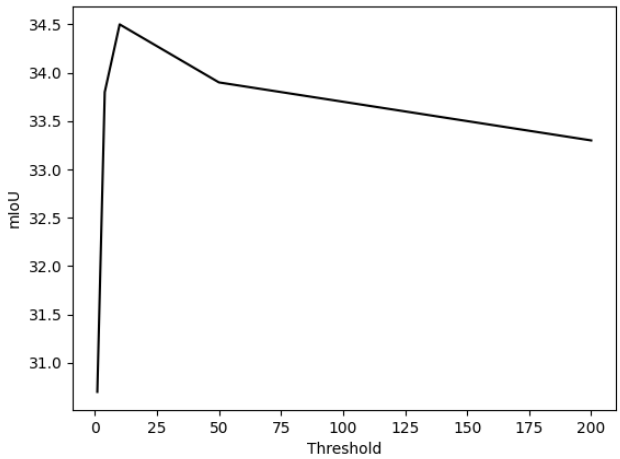


Figure 6.11: Effect of the radius.

6.4 Experiments

6.4.1 Environment Setup

We use four challenging datasets to evaluate our method:

- **SemanticKITTI** (*Behley et al.*, 2019d) provides pixel-wise semantic labels for the entire KITTI Odometry Benchmark (*Geiger et al.*, 2012b) with more than 20000 scans which are divided into 22 sequences. We use its training set for training (sequences 00-10 without sequence 08), its validation set (sequence 08) for the ablation study, and its test set (sequences 10-21) for comparison with state-of-the-art methods.

- **SemanticPOSS** (*Pan et al.*, 2020a) contains about 3000 scans at Peking University, which are divided into 6 equal subsets. For the experiments, we use the 3rd subset for evaluation and the others for training as in *Pan et al.* (2020a). There are usually more moving and small objects in the campus scenarios, making it more difficult compared to urban environments.

- **nuScenes** (*Caesar et al.*, 2020) includes about 40000 annotated scans captured in 900 scenes, where 750 scenes are for training and the others for validation. As recommended[1], we merge similar classes and remove rare classes.

- **Pandaset**[2] contains about 16000 LiDAR scans at 2 routes in Silicon Valley. Two LiDAR sensors have been used for recording the data, a spinning LiDAR and a solid-state LiDAR. For the experiments, the data from the spinning LiDAR is used. We use 30% of the data for evaluation and the rest for training. Similar to the nuScenes dataset, we merge similar classes and remove rare classes.

[1]The nuScenes LiDAR segmentation task is available at https://www.nuscenes.org/lidar-segmentation.
[2]https://scale.com/open-datasets/pandaset.

	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	mIoU
Pointnet (*Qi et al.*, 2017b)	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7	14.6
Pointnet++ (*Qi et al.*, 2017d)	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9	20.1
SPGraph (*Landrieu and Simonovsky*, 2018)	68.3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5	20.0
SPLATNet (*Su et al.*, 2018)	66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0	22.8
TangentConv (*Tatarchenko et al.*, 2018)	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2	35.9
DeepLabV3+(*Chen et al.*, 2018)	78.4	13.6	9.5	9.5	10.4	17.5	22.0	0.4	88.5	54.5	66.7	9.7	77.9	39.1	72.0	39.9	60.0	23.4	36.1	38.4
PSPNet(*Li et al.*, 2018)	79.6	25.0	26.4	17.5	24.0	34.1	28.4	7.3	90.2	58.2	70.2	19.9	79.7	43.5	74.2	43.2	61.2	23.1	37.5	44.4
DenseASPP(*Yang et al.*, 2018)	78.1	20.5	18.2	20.0	16.6	27.8	28.9	5.7	88.5	53.3	67.5	9.3	76.3	39.6	70.0	36.8	57.7	15.9	32.4	40.2
SqueezeSeg (*Wu et al.*, 2018b)	68.8	16.0	4.1	3.3	3.6	12.9	13.1	0.9	85.4	26.9	54.3	4.5	57.4	29.0	60.0	24.3	53.7	17.5	24.5	29.5
SqueezeSeg + CRF (*Wu et al.*, 2018b)	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	30.8
SqueezeSegV2 (*Wu et al.*, 2019b)	81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3	39.7
SqueezeSegV2 + CRF (*Wu et al.*, 2019b)	82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0	39.6
RangeNet21 (*Milioto et al.*, 2019b)	85.4	26.2	26.5	18.6	15.6	31.8	33.6	4.0	91.4	57.0	74.0	26.4	81.9	52.3	77.6	48.4	63.6	36.0	50.0	47.4
RangeNet53 (*Milioto et al.*, 2019b)	86.4	24.5	32.7	**25.5**	22.6	36.2	33.6	4.7	**91.8**	**64.8**	**74.6**	**27.9**	84.1	**55.0**	78.3	50.1	64.0	38.9	52.2	49.9
MINet	85.2	38.2	32.1	29.3	23.1	47.6	46.8	24.5	90.5	58.8	72.1	25.9	82.2	49.5	78.8	52.5	65.4	37.7	55.5	52.4
RPointNet++	73.3	13.0	5.4	11.8	8.3	6.4	15.5	2.1	86.3	40.1	60.1	7.2	61.7	32.1	55.8	13.1	51.6	4.2	14.7	29.6
RSCNN	79.8	19.8	11.2	15.8	14.3	15.1	20.5	8.8	87.1	41.8	64.7	8.7	72.2	37.9	68.0	28.4	58.0	13.1	31.3	36.7
RPointConv	79.5	19.0	12.7	13.8	10.7	14.9	18.2	5.8	87.8	46.6	66.6	7.3	73.2	40.1	69.4	30.9	59.3	14.1	32.1	36.9
UnPNet	70.8	38.2	31.8	20.3	22.5	49.0	45.8	14.3	91.4	61.6	73.6	19.2	82.4	50.8	77.7	51.6	65.3	34.9	53.6	51.0
UnPNet + *k*-NN	**90.4**	**43.8**	**36.1**	20.4	**23.1**	**54.3**	**54.2**	**14.4**	91.4	62.0	74.2	18.9	**86.3**	54.6	**80.5**	**59.4**	**66.3**	**48.7**	**58.6**	**54.6**

Table 6.4: Evaluation results on the SemanticKITTI dataset. MIN is the method proposed in Chapter 4.

	barrier	bicycle	bus	car	cons_vehicle	motorcycle	pedestrian	traffic_cone	trailer	truck	driv_surf	other_flat	sidewalk	terrain	manmade	vegetation	mIoU
DeepLabV3 (*Chen et al.*, 2018)	50.5	4.9	58.6	60.3	10.6	7.5	21.6	17.2	35.1	46.2	88.1	47.2	55.7	59.2	69.1	69.4	43.8
DenseASPP (*Yang et al.*, 2018)	52.7	6.2	69.0	67.1	18.5	32.6	24.3	16.9	39.7	58.0	87.4	43.1	57.1	58.6	69.8	70.4	48.2
PSPNet(*Zhao et al.*, 2017)	56.9	8.9	69.8	68.9	**23.6**	36.9	26.5	18.9	**42.3**	58.7	88.0	44.5	58.5	59.6	71.5	71.6	50.3
SqueezeSegV1 (*Wu et al.*, 2018b)	15.0	0.3	4.5	25.8	0.0	0.5	3.1	5.3	2.5	9.4	68.3	11.2	23.3	42.1	45.6	40.1	18.6
SqueezeSegV2 (*Wu et al.*, 2019b)	44.3	2.7	62.2	68.0	11.2	19.3	7.6	12.1	25.3	44.8	84.8	29.8	51.6	56.6	64.4	67.8	40.8
RangeNet21 (*Milioto et al.*, 2019b)	61.4	3.4	72.9	76.4	17.2	23.5	31.5	19.3	35.8	59.8	92.3	54.6	66.5	68.1	76.9	76.6	52.3
RangeNet53 (*Milioto et al.*, 2019b)	59.8	2.7	62.6	73.5	14.1	21.6	28.3	13.9	34.5	58.3	90.8	53.8	61.7	64.5	74.8	75.0	49.4
UnPNet	**61.2**	5.9	**77.7**	73.2	21.9	34.7	38.5	25.7	40.3	62.9	92.3	61.6	66.7	67.7	78.7	78.9	55.5
UnPNet + *k*-NN	61.0	**6.3**	**77.7**	**78.4**	21.9	**37.0**	42.5	**30.5**	41.7	**65.8**	**93.8**	**62.2**	**66.9**	**68.1**	**80.6**	**80.0**	**57.2**

Table 6.5: Evaluation results on the nuScenes dataset.

As for the evaluation metric, we use the standard mean intersection over union (mIoU) metric (*Everingham et al.*, 2015) over all classes. For a fair comparison, all experiments are performed with a single GPU.

6.4.2 Ablation Study

The ablation study is conducted on the SemanticKITTI dataset.

6.4.2.1 Effect of hyperparameters

We first explore the influence of the size k of the 2D search region in the reformulated grouping layer. For the study, we use the reformulated PointNet++ (RPointNet++) with an input resolution of 64×512. The results are shown in Table 6.1(a). We can see that the performance improves and the speed slows down with the search size k increasing. However, the improvement from $k = 5$ to $k = 7$ is not as large as that from $k = 3$ to $k = 5$, which indicates that the local region with $k = 5$ already includes the most important neighboring points. Considering the trade-off between effectiveness and efficiency, we set $k = 5$ in the following experiments.

	Vegetation	Ground	Road	Lane Line	Road Mark	Sidewalk	Car	Truck	Motorcycle	Bus	Bicycle	Pedestrian	Pylons	Signs	Cones	Const-Signs	Building	mIoU
DeepLabV3 (*Chen et al.*, 2018)	50.4	22.8	69.2	11.9	9.7	35.5	65.6	11.6	0.1	3.9	0.2	3.5	2.2	19.2	4.2	11.5	58.4	22.3
DenseASPP (*Yang et al.*, 2018)	54.6	21.2	66.0	9.9	9.3	33.2	64.9	18.8	0.0	17.6	0.2	5.0	0.3	21.2	2.7	4.6	62.5	23.1
PSPNet (*Zhao et al.*, 2017)	45.9	18.1	65.4	8.4	6.8	28.1	60.2	15.6	0.0	6.2	0.2	2.5	0.0	19.0	0.9	4.5	55.2	19.8
SqueezeSegV1 (*Wu et al.*, 2018b)	40.0	16.5	66.7	9.4	6.3	19.2	52.4	9.6	0.0	3.9	0.3	2.6	0.1	15.5	1.2	0.0	39.4	16.6
SqueezeSegV2 (*Wu et al.*, 2019b)	61.0	33.0	76.4	18.6	13.5	43.5	73.5	30.8	0.0	21.8	1.7	4.9	1.1	20.5	3.3	3.7	65.4	27.8
RangeNet21 (*Milioto et al.*, 2019b)	65.9	35.3	81.1	26.6	20.0	47.1	75.0	28.6	0.1	**25.7**	1.5	12.0	1.6	34.2	4.9	19.0	72.4	32.4
RangeNet53 (*Milioto et al.*, 2019b)	64.1	36.3	80.8	25.1	19.8	48.8	76.0	30.7	**1.0**	17.2	2.0	9.8	6.3	32.7	6.7	13.0	71.0	31.8
UnPNet	75.4	40.7	82.4	26.6	19.9	49.8	76.4	31.1	0.7	17.5	10.4	29.5	28.6	44.4	14.5	19.5	79.6	38.1
UnPNet + *k*-NN	**78.2**	**41.6**	**82.5**	**29.3**	**21.7**	**52.0**	**78.9**	**31.9**	0.9	17.7	**12.1**	**36.7**	**40.4**	**63.7**	**19.5**	**24.9**	**83.8**	**42.1**

Table 6.6: Evaluation results on the Pandaset dataset.

	person	rider	car	trunk	plants	traffic sign	pole	building	fence	bike	road	mIoU
DeepLabV3 (*Chen et al.*, 2018)	9.6	4.8	15.3	9.5	40.3	5.3	3.1	41.2	11.9	20.0	62.5	20.3
DenseASPP (*Yang et al.*, 2018)	11.6	5.7	20.4	10.8	46.4	5.2	4.9	46.8	7.7	20.6	62.0	22.0
PSPNet (*Zhao et al.*, 2017)	9.0	4.8	17.6	10.3	44.1	5.1	3.2	45.3	5.9	20.0	63.0	20.8
SqueezeSegV1 (*Wu et al.*, 2018b)	5.5	0.0	8.7	3.4	39.1	2.4	2.5	34.5	7.6	18.4	62.5	16.8
SqueezeSegV2 (*Wu et al.*, 2019b)	**18.4**	11.2	34.9	**15.8**	56.3	**11.0**	4.5	47.0	25.5	32.4	**71.3**	29.8
RangeNet21 (*Milioto et al.*, 2019b)	9.8	7.8	**39.9**	8.0	55.8	7.0	2.9	50.3	19.2	32.3	63.8	27.0
RangeNet53 (*Milioto et al.*, 2019b)	10.0	6.2	33.4	7.3	54.2	5.5	2.6	49.9	18.4	28.6	63.5	25.4
UnPNet	11.3	12.1	36.8	10.6	62.3	6.9	4.2	60.4	20.6	35.4	65.6	29.7
UnPNet + *k*-NN	17.7	**17.2**	39.2	13.8	**67.0**	9.5	**5.8**	**66.9**	**31.1**	**40.5**	68.4	**34.3**

Table 6.7: Evaluation results on the SemanticPOSS dataset.

Then, we evaluate the effect of different input resolutions, and we summarize the results in Table 6.1(b). We can observe that increasing the input resolution from 64×512 to 64×1024 improves the accuracy but at the cost of higher inference time. However, when the resolution changes from 64×1024 to 64×2048, the accuracy degrades. Since we kept k constant, increasing the resolution decreases the receptive field. Hence, k needs to be increased when the resolution increases. For a good trade-off between effectiveness and efficiency, we set the input resolution to 64×512 if not mentioned otherwise.

We also evaluate the impact of the stride of the uniform sampling, i.e., $H' = \frac{H}{S}$ and $W' = \frac{W}{S}$. The results are shown in Table 6.1(c). The accuracy is similar for the sampling stride 1 and 2, but using stride 2 is more efficient. When the sampling stride is set to 4, the accuracy drops significantly since the sampling becomes too sparse. We use therefore sampling stride 2.

As for the impact of the radius, we vary it between 1m and 200m. The results are shown in Figure 6.11. The mIoU increases until a radius of 10m and then slightly decreases. We use 10m as the default threshold.

6.4.2.2 Comparison between reformulated and original methods

Finally, we compare the original models with their reformulated versions. The three baseline methods are PointNet++ (*Qi et al.*, 2017d), SpiderCNN (*Xu et al.*, 2018) and PointConv (*Wu et al.*, 2019c). We display the results in Table 6.2. We can see that the reformulated point-based methods achieve a much better performance than the original baselines. The improvement is about 10% for all baselines in terms of both accuracy and mIoU. For SpiderCNN (*Xu et al.*, 2018) and PointConv (*Wu et al.*, 2019c), the mIoU is even improved by about 15%. These results prove the effectiveness of the reformulated point-based methods. Apart from the mIoU and accuracy, the efficiency is also much better. We can see that the reformulated methods are more than 300× faster compared with the original ones. This

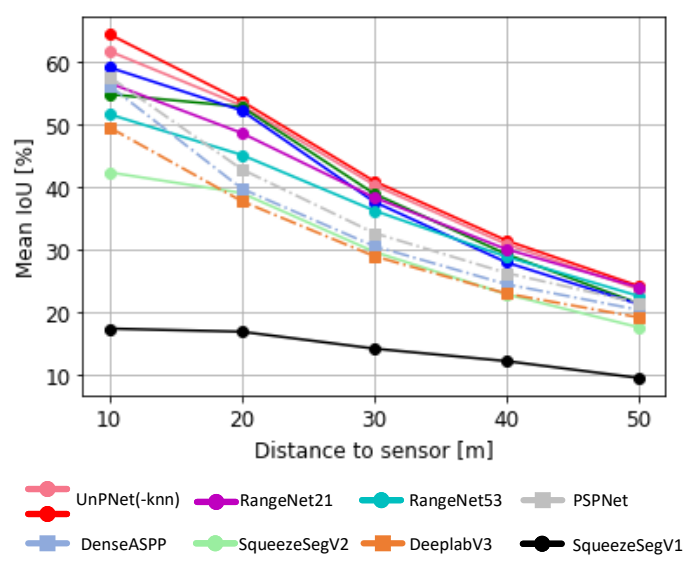
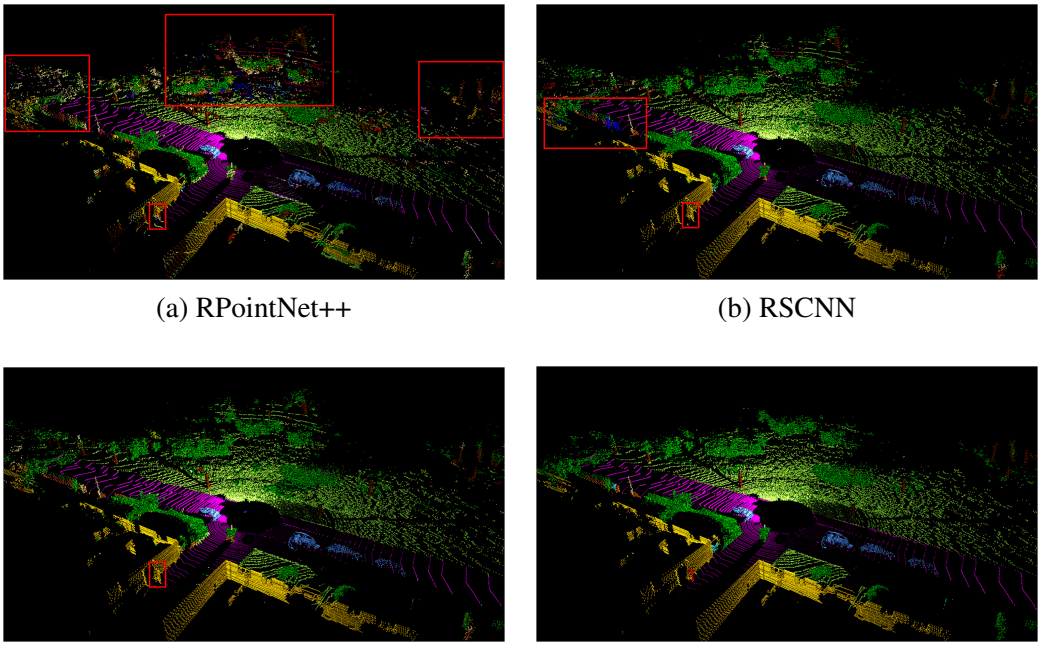


Figure 6.12: Mean IoU with regard to the distance of the points to the LiDAR sensor, evaluated on the nuScenes dataset (*Caesar et al.*, 2020). The dashed lines denote image-based methods while solid lines represent projection-based methods. For UnPNet, we show both the results with k-NN (red) and without k-NN post-processing (light red).



(a) RPointNet++ (b) RSCNN

(c) RPointConv (d) Ground Truth

Figure 6.13: Qualitative results of RPointNet++, RSCNN, and RPointConv. We highlight wrong predictions by the red boxes. RPointNet++ makes wrong predictions for distant points due to the weak aggregation capability of the used pooling operations. RPointConv achieves better results than RSCNN, demonstrating the effectiveness of density information.

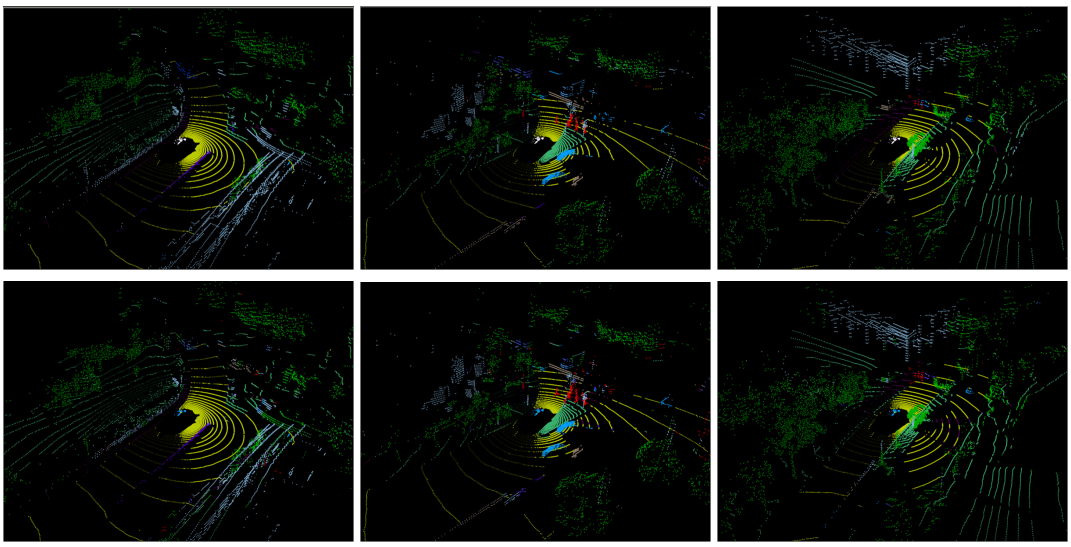


Figure 6.14: Qualitative results of UnPNet for the nuScenes dataset. Top: Ground Truth, Bottom: Prediction.

is because the reformulated methods utilize the projection space such that the 3D operations can be performed much more efficiently. This demonstrates how point-based methods can be improved by reformulating them. Figure 6.13 shows some qualitative results of the three reformulated versions.

6.4.2.3 Ablation study for UnPNet

After having discussed the reformulated point-based networks, we now evaluate UnPNet and the results are shown in Table 6.3. Although UnPNet uses the reformulated feature propagation from RPointConv, it outperforms RPointConv by a large margin. While the inference time is very similar, the mIoU is much higher for UnPNet compared to RPointConv. This demonstrates that leveraging reformulated 3D point-based operations and 2D image-based operations achieves a good performance. We also analyze the impact of the edge supervision. Without edge supervision, the mIoU decreases by 0.5%. The k-NN post-processing increases the accuracy.

6.4.3 Comparison with State-of-the-art

We compare the proposed reformulated networks RPointNet++, RSCNN, and RPointConv as well as UnPNet with other methods. For a comprehensive comparison, we selected point-based methods (Pointnet (*Qi et al.*, 2017b), Pointnet++ (*Qi et al.*, 2017d), SPGraph (*Landrieu and Simonovsky*, 2018), SPLATNet (*Su et al.*, 2018), TangentConv (*Tatarchenko et al.*, 2018)), image-based methods (DeepLabV3 (*Chen et al.*, 2018), DenseASPP (*Yang et al.*, 2018), PSPNet (*Zhao et al.*, 2017)) and projection-based methods (SqueezeSeg (*Wu et al.*, 2018b, 2019b), RangeNet (*Milioto et al.*, 2019b)).

We first show the results for the SemanticKITTI test dataset in Table 6.4. While the reformulated point-based methods RPointNet++, RSCNN, and RPointConv outperform the corresponding baselines PointNet++, SCNN, and PointConv as on the validation set, they do not achieve the accuracy of state-of-the-art approaches. However, as we discussed, the general concept of reformulating point-based methods can also be used to develop new architectures by leveraging reformulated 3D

operations and 2D CNNs. This is done by the proposed UnPNet and we can see that it outperforms RPointNet++, RSCNN, and RPointConv by a large margin. Using k-NN as in (*Milioto et al.*, 2019b) for post-processing improves the accuracy further. Compared to the other approaches, UnPNet performs in particular well for small objects like bicycle, person, or motorcyclist.

The results for the nuScenes dataset are shown in Table 6.5. Similar to the SemanticKITTI dataset, UnPNet outperforms the other methods by a large margin. Furthermore, it achieves the best performance for almost all classes. In Figure 6.12, we also report the mIoU based on the distance to the sensor. We can see that as the points are more distant to the LiDAR sensor, the accuracy decreases as expected. Nevertheless, UnPNet outperforms the other methods for all distances. We can also observe that the k-NN post-processing mainly improves the accuracy at the close distances. In Figure 6.14, we show some qualitative results for UnPNet.

For the Pandaset dataset, UnPNet can successfully detect some small objects that other methods hardly recognize like bicycle or cones which can be seen in Table 6.6. The overall performance of our method is more than 10% higher compared with the other methods. Finally, we present the results for the SemanticPOSS dataset in Table 6.7. Our method also outperforms the other methods and improves mIoU by about 4%. In summary, UnPNet achieves the best performance on all four datasets. This shows the robustness of UnPNet.

6.5 Summary

In this chapter, we proposed a new paradigm to reformulate point-based methods so that they operate in the projection space of a LiDAR sensor. As examples, we used three point-based approaches and demonstrated that reformulated point-based methods achieve a better segmentation accuracy and efficiency than the original point-based methods. Furthermore, we proposed a new architecture named Unprojection Network (UnPNet) which combines the benefits of both point-based and projection-based methods. On one hand, it extracts features efficiently like projection-based methods and fuses the context information from different 2D scales. On the other hand, it exploits the full 3D structure as point-based methods for down- and upsampling. In addition, auxiliary edge supervision is utilized, which is infeasible for point-based methods. We evaluated the approach on four challenging datasets for semantic LiDAR point cloud segmentation and showed that the proposed concept of reformulating 3D point-based operations allows to design new architectures that outperform point- and projection-based approaches. As our approach of reformulating point-based operations is generic, it can be also used to make point-voxel methods like PV-RCNN (*Shi et al.*, 2020b) more efficient by reformulating the point-based operations within the network. We will explore this in the future work.

CHAPTER 7

TFNet: Exploiting Temporal Cues for Fast and Accurate LiDAR Semantic Segmentation

In previous chapters, we mainly focus on conducting semantic segmentation on the current LiDAR scan. However, in realistic autonomous driving, the previous LiDAR scans are accessible which are usually ignored by previous works. Another challenge is a "many-to-one" problem for a single LiDAR scan caused by the range image's limited horizontal and vertical angular resolution. As a result, around 20% of the 3D points can be occluded. In this chapter, we present TFNet, a projection-based LiDAR semantic segmentation method that utilizes temporal information to address this issue. Specifically, we incorporate a temporal fusion layer to extract useful information from previous scans and integrate it with the current scan. We then design a max-voting-based post-processing technique to correct false predictions, particularly those caused by the "many-to-one" issue. We evaluated the approach on two benchmarks and demonstrated that the post-processing technique is generic and can be applied to various networks.

Contents

7.1 Introduction

As mentioned above, the range view representation suffers from a boundary-blurring effect (*Milioto et al.*, 2019c; *Zhao et al.*, 2021). This problem exists mainly because of the limited horizontal and vertical angular resolution: more than one point will be projected to the same range image pixel

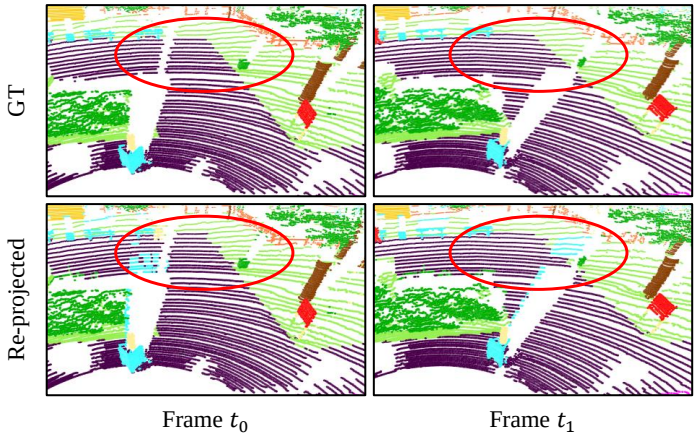


Figure 7.1: Projection-based methods suffer from the "many-to-one" problem where multiple 3D points with the same angle are mapped to a single range pixel. This can cause distant points to receive erroneous predictions (marked by the red circles) from nearby points when the range image is re-projected to 3D. Moreover, the occluded points in t_0 are visible in t_1, which can help correct the predictions.

when these points share the same vertical and horizontal angle. This is also referred to as the "many-to-one" problem (*Zhao et al.*, 2021). Considering that the projection computes distant points first and near points later (*Milioto et al.*, 2019c), the distant points will be occluded by the near points. Hence, during the re-projection of the range image into three-dimensional coordinates, a necessary procedure for projection-based approaches, distant points will be assigned the same predicted label as the overlapping closer points. As shown in Figure 7.1, this introduces errors for the distant points. In fact, we measure the impact of this phenomenon on SemanticKITTI (*Behley et al.*, 2019c, 2021a). If the range image has a size of $(H, W) = (64, 2048)$, over 20% of the 3D points are occluded in the range image, i.e., more than one point is projected to the same pixel. As shown in Table 7.3, this results in a substantial degradation of the accuracy if it is not addressed by an additional post-processing step. Therefore, various post-processing approaches like k-NN (*Milioto et al.*, 2019c), CRF (*Wu et al.*, 2018a) or NLA (*Zhao et al.*, 2021) have been proposed. For instance, NLA (*Zhao et al.*, 2021) assigns the prediction of the nearest point to the occluded points.

In this work, we propose to incorporate temporal information to address the "many-to-one" challenge for LiDAR semantic segmentation. This is inspired by human visual perception, where temporal information is crucial for understanding object motion and identifying occlusions. This is also observed in LiDAR semantic segmentation, where heavily occluded points can be captured from adjacent range image scans, as shown in Figure 7.1. Based on this intuition, we exploit the temporal relations of features in the range map via cross-attention (*Vaswani et al.*, 2017a; *Li et al.*, 2022; *Gao et al.*, 2022). As for the inference stage, we propose a max-voting-based post-processing scheme that effectively reuses the predictions of past frames. To this end, we transform the previous scans with predicted semantic class labels into the current ego car coordinate frame and then obtain the final segmentation by aggregating the predictions within the same voxel by max-voting.

In summary, we make the following three contributions:

- We propose TFNet, a projection-based LiDAR semantic segmentation method. It utilizes a

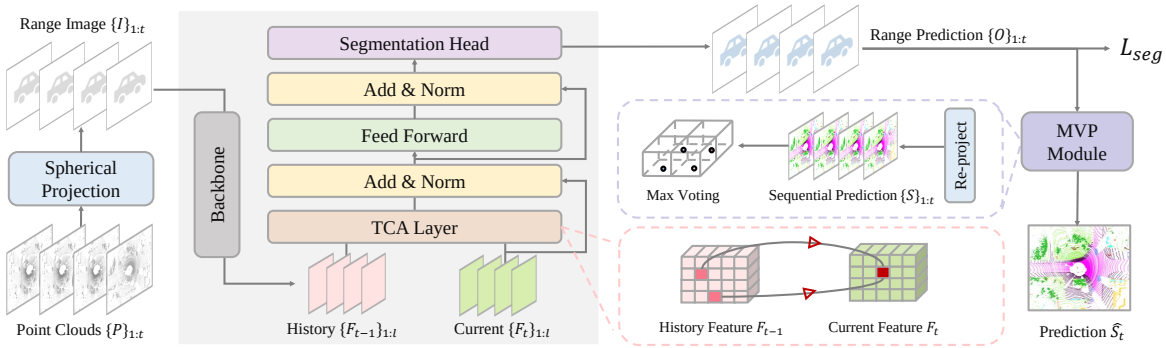


Figure 7.2: Architecture of TFNet. For a point cloud P_t, TFNet projects it onto range images I_t. It then uses a segmentation backbone to extract multi-scale features $\{F_t\}_{1:l}$, a **T**emporal **C**ross-**A**ttention (**TCA**) layer to integrate past features $\{F_{t-1}\}_{1:l}$, and a segmentation head to predict projection based logits O_t. In inference, it refines the re-projected prediction S_t by aggregating the current and past temporal predictions $\{S\}_{1:t}$ by a **M**ax-**V**oting-based **P**ost-processing (**MVP**) strategy.

temporary cross-attention layer, which extracts informative features from previous LiDAR scans and combines them with current range features, to compensate for occluded objects.

- We design a temporal-based post-processing method to solve the "many-to-one" mapping issue in range images. Compared with previous post-processing steps, our method achieves better performance, which is verified for various networks.

- We evaluate the proposed method on two public benchmarks, namely SemanticKITTI (*Behley et al.*, 2019c) and SemanticPOSS (*Pan et al.*, 2020b), where our method achieves a good trade-off between accuracy and inference time.

7.2 Method

7.2.1 Network Overview

The overview of our proposed network is illustrated in Figure 7.2. Our proposed network takes as input a point cloud P comprising N points represented by 3D coordinates x, y, z, and intensity i. The point cloud is projected onto a range image I of size $H \times W \times 5$ using a spherical projection technique employed in previous works (*Milioto et al.*, 2019c; *Wu et al.*, 2018a). Here, H and W represent the height and width of the image, and the last dimension includes coordinates (x, y, z), range $r = \sqrt{x^2 + y^2 + z^2}$, and intensity i. Next, we feed the range image into our backbone model to obtain multi-scale features F with resolutions $\{1, 1/2, 1/4, 1/8\}$. We employ a **T**emporal **C**ross-**A**ttention (**TCA**) layer to integrate spatial features from history frame. The aggregated features are then fed to the segmentation head, which predicts the projection-based semantic segmentation logits O. For inference, we re-project the 2D semantic segmentation prediction to a 3D point-wise prediction S. Subsequently, we propose a **M**ax-**V**oting-based **P**ost-processing (**MVP**) strategy to refine the current prediction S_t by aggregating previous predictions. We describe the key components of our network in the following sections.

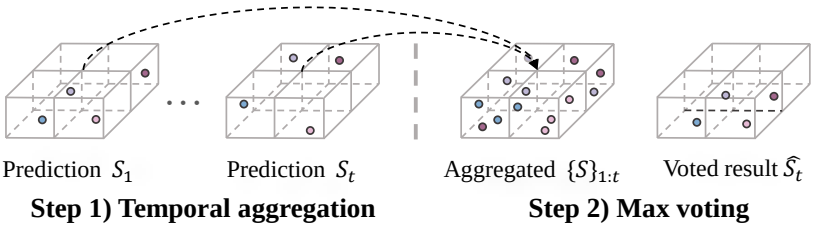


Figure 7.3: Illustration of the max voting post-processing strategy.

7.2.2 Temporal Cross Attention

Although the range image suffers from the "many-to-one" issue, the occluded points can be captured from adjacent scans. This observation motivates us to incorporate sequential scans into both the training and inference stages. First, we discuss how sequential data can be exploited during the training stage.

Inspired by the notable information extraction ability of the attention mechanism (*Vaswani et al.*, 2017a) verified by various other works (*Xie et al.*, 2021; *Li et al.*, 2022; *Wang et al.*, 2023; *Ma et al.*, ICCV), we use the cross-attention mechanism to model the temporal connection between the previous range feature F_{t-1} and the current range feature F_t.

The attended value is computed by:

$$\mathbf{x}_{in} = \text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot K^{\mathsf{T}}}{\sqrt{d_f}}\right) V, \tag{7.1}$$

where Q, K, V are obtained by $Q = Linear_q(F_t)$, $K = Linear_k(F_{t-1})$, $V = Linear_v(F_{t-1})$, and d_f is the dimension of the range features.

We integrate a 3×3 convolution into the feed-forward module to encode positional information as in *Xie et al.* (2021) as well as a residual connection (*He et al.*, 2016b). The **feed-forward module** is defined as follows:

$$\mathbf{x}_{out} = \text{MLP}(\text{GELU}(\text{Conv}_{3\times3}(\text{MLP}(\mathbf{x}_{in})))) + \mathbf{x}_{in}. \tag{7.2}$$

The TCA module effectively exploits temporal dependencies in two ways. First, instead of using multiple stacked range features (*Chen et al.*, 2021c), our method extracts temporal information from the previous range features. This not only reduces computational costs but also minimizes the influence of moving objects, which can introduce noise into the data. Secondly, we only utilize the fusion module on the last feature level, which significantly decreases computation complexity. Previous works (*Li et al.*, 2022; *Gao et al.*, 2022) have shown that the attention at shallower layers is not effective.

7.2.3 Max-voting-based Post-processing

While temporal cross attention exploits temporal information at the feature level, it does not resolve the "many-to-one" issue during the re-projection process of a projection-based method, which causes occluded far points to inherit the predictions of near points. We thus propose a max-voting-based post-processing (MVP) strategy, which is motivated by the observation that occluded points will be

	Input	mean-IoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
PointNet (*Qi et al.*, 2017a)	Point	14.6	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7
PointNet++ (*Qi et al.*, 2017c)		20.1	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9
LatticeNet (*Luo et al.*, 2020b)		52.9	92.9	16.6	22.2	26.6	21.4	35.6	43.0	46.0	90.0	59.4	74.1	22.0	88.2	58.8	81.7	63.6	63.1	51.9	48.4
RandLA-Net (*Hu et al.*, 2020b)		53.9	94.2	26.0	25.8	40.1	38.9	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7
KPConv (*Thomas et al.*, 2019b)		58.8	**96.0**	30.2	42.5	33.4	44.3	61.5	61.6	11.8	88.8	61.3	72.7	31.6	90.5	64.2	84.8	69.2	69.1	56.4	47.4
BAAF-Net (*Qiu et al.*, 2021)		59.9	95.4	31.8	35.5	**48.7**	46.7	49.5	55.7	33.0	90.9	62.2	74.4	23.6	89.8	60.8	82.7	63.4	67.9	53.7	52.0
UnPNet	Range	54.6	90.4	43.8	36.1	20.4	23.1	54.3	54.2	14.4	91.4	62.0	74.2	18.9	86.3	54.6	80.5	59.4	66.3	48.7	58.6
MINet		55.2	90.1	41.8	34.0	29.9	23.6	51.4	52.4	25.0	90.5	59.0	72.6	25.8	85.6	52.3	81.1	58.1	66.1	49.0	59.9
FIDNet (*Zhao et al.*, 2021)		59.5	93.9	54.7	48.9	27.6	23.9	62.3	59.8	23.7	90.6	59.1	75.8	26.7	88.9	60.5	84.5	64.4	69.0	53.3	62.8
Meta-RangeSeg (*Wang et al.*, 2022b)		61.0	93.9	50.1	43.8	43.9	43.2	63.7	53.1	18.7	90.6	64.3	74.6	29.2	91.1	64.7	82.6	65.5	65.5	56.3	64.2
KPRNet (*Kochanov et al.*, 2020b)		63.1	95.5	54.1	47.9	23.6	42.6	65.9	65.0	16.5	**93.2**	**73.9**	**80.6**	30.2	91.7	68.4	**85.7**	69.8	**71.2**	58.7	64.1
Lite-HDSeg (*Razani et al.*, 2021)		63.8	92.3	40.0	55.4	37.7	39.6	59.2	71.6	**54.1**	93.0	68.2	78.3	29.3	91.5	65.0	78.2	65.8	65.1	59.5	67.7
CENet (*Cheng et al.*, 2022)		64.7	91.9	58.6	50.3	40.6	42.3	68.9	65.9	43.5	90.3	60.9	75.1	31.5	91.0	66.2	84.5	69.7	70.0	**61.5**	67.6
RangeViT (*Ando et al.*, 2023)		64.0	95.4	55.8	43.5	29.8	42.1	63.9	58.2	38.1	93.1	70.2	80.0	**32.5**	**92.0**	**69.0**	85.3	70.6	**71.2**	60.8	64.7
LENet (*Ding*, 2023)		64.5	93.9	57.0	51.3	44.3	44.4	66.6	64.9	36.0	91.8	68.3	76.9	30.5	91.2	66.0	83.7	68.3	67.8	58.6	63.2
TFNet (Ours)		**66.1**	94.3	**60.7**	**58.5**	38.4	**48.4**	**74.3**	**72.2**	35.5	90.6	68.5	75.3	29.0	91.6	67.3	83.8	**71.1**	67.0	60.8	**68.7**

Table 7.1: Performance comparison on SemanticKITTI test set. "MINet" is the method in Chapter 4 while "UnPNet" is the method in Chapter 6.

visible in the adjacent scans as shown in Figure 7.1. As verified in Table 7.5, MVP is generic and can be added to various networks.

Temporal scan alignment. To initiate post-processing, it is essential to align a series of past LiDAR scans $(P_1, ..., P_t)$ to the viewpoint (i.e., coordinate frame) of P_t. The alignment is accomplished by utilizing the estimated relative pose transformations T^j_{j-1} between the scans P_{j-1} and P_j. These transformation matrices can be acquired from an odometry estimation approach such as SuMa++ (*Chen et al.*, 2019b). The relative transformations between the scans $(T^2_1, ..., T^t_{t-1})$ are represented by transformation matrices of $T^j_{j-1} \in \mathbb{R}^{4\times4}$. Further, we denote the j^{th} scan transformed to the viewpoint of P_t by

$$P^{j\to t} = \{T^t_j p_i\}_{p_i \in P_j} \quad \text{with } T^t_j = \prod_{k=j+1}^{t} T^k_{k-1}. \tag{7.3}$$

Sparse grid max voting. After applying the transformations, we aggregate the aligned scans. We quantize the aggregated scans into a voxel grid with a fixed resolution δ. In each grid, we use the max-voting strategy to use the most frequently predicted class label to represent the semantics of all points in the grid. We illustrate this process in Figure 7.3 and evaluate the impact of the grid size in Figure 7.6.

To save computation and memory, we store only the non-empty voxels. This sparse representation allows our method to handle large scenes.

Sliding window update. We initialize a sliding window $W_{t-L+1:t}$ with the length of L to store the scans and use a FIFO (First In First Out) strategy to update the points falling in each grid. When the LiDAR sensor obtains a new point cloud scan, we add it to this sliding window and remove the oldest scan. We do not use different weights across frames due to the uncertain occlusion problem.

7.3 Experiments

Datasets and evaluation metrics. We use two benchmark datasets, SemanticKITTI (*Behley et al.*, 2019c) and SemanticPOSS (*Pan et al.*, 2020b), to evaluate our proposed method. SemanticKITTI (*Behley et al.*, 2019c) is a popular benchmark for LiDAR-based semantic segmentation in driving scenes. It contains 19,130 training frames, 4,071 validation frames, and 20,351 test frames. Each point in the dataset is provided with a semantic label of 19 classes for semantic segmentation. We also evaluate our dataset on the SemanticPOSS (*Pan et al.*, 2020b) dataset, which contains 2988 scenes for training and testing. For evaluation, we follow previous works (*Cheng et al.*, 2022; *Kong et al.*, 2023b; *Zhao et al.*, 2021; *Wu et al.*, 2018a), utilizing the class-wise Intersection over Union (IoU) and mean IoU (mIoU) metrics to evaluate and compare with others.
Implementation details. While we use CENet (*Cheng et al.*, 2022) as the main baseline method, our method demonstrates robust generalization across various backbones as shown in the following experiments. We train the proposed method using the Stochastic Gradient Descent (SGD) optimizer and set the batch size to 8 and 4 for SemanticKITTI and SemanticPOSS, respectively. We follow the baseline method (*Cheng et al.*, 2022) to supervise the training with a weighted combination of cross-entropy, Lovász softmax loss (*Berman et al.*, 2018b), and boundary loss (*Bokhovkin and Burnaev*, 2019). The weights for the loss terms are set to $\beta_1 = 1.0$, $\beta_2 = 1.5$, $\beta_3 = 1.0$, respectively. All the models are trained on GeForce RTX 3090 GPUs. The inference latency is measured using a single GeForce RTX 3090 GPU. The backbone is trained from scratch on all the datasets.

7.3.1 Comparison with State-of-the-arts

Quantitative results on SemanticKITTI. Table 7.1 reports comparisons with representative models on the SemanticKITTI test set. Our method outperforms all projection-based methods, including CNN-based architectures (*Cheng et al.*, 2022; *Zhao et al.*, 2021; *Li et al.*, 2021a) and Transformer-based architectures (*Ando et al.*, 2023) in terms of mean IoU. CENet (*Cheng et al.*, 2022) uses test time augmentation to improve the performance. For a fair comparison with previous methods (*Milioto et al.*, 2019c; *Li et al.*, 2021a), we do not use test time augmentation.
Quantitative results on SemanticPOSS. We present our performance on the SemanticPOSS test set (*Pan et al.*, 2020b) in Table 7.2. Our approach outperforms all approaches in terms of mean IoU. Compared to CENet, our method shows a significant advantage in predicting small objects such as traffic signs, poles, and cones/stones.
Inference time comparison. We visualize the inference time and mIoU of popular methods in Figure 7.4. The results show that projection-based methods are faster than point, voxel, or hybrid methods. We measured the inference time of all the methods on the same hardware with a GeForce RTX 3090 GPU for a fair comparison.

7.3.2 Ablation Analysis

Effect of the temporal post-processing. Table 7.3 compares the proposed post-processing method with other post-processing approaches on the SemanticKITTI validation set. Using a CRF for post-processing has been used by SequeezeSegv2 (*Wu et al.*, 2019a). We train the network with CRF from scratch using the same training pipeline as our method. The k-Nearest Neighbor (k-NN) method (*Milioto et al.*, 2019c) is the most popular post-processing method. It is widely used in

	Sq.Seg (*Wu et al.*, 2018a)	Sq.SegV2 (*Wu et al.*, 2019a)	RangeNet (*Milioto et al.*, 2019c)	MINet (*Li et al.*, 2021a)	FIDNet (*Zhao et al.*, 2021)	CENet (*Cheng et al.*, 2022)	TFNet (Ours)
person	6.8	43.9	57.3	62.4	72.2	**75.5**	72.4
rider	0.6	7.1	4.6	12.1	**23.1**	22.0	20.5
car	6.7	47.9	35.0	63.8	72.7	77.6	**77.7**
truck	4.0	18.4	14.1	22.3	23.0	**25.3**	24.8
plants	2.5	40.9	58.3	68.6	68.0	**72.2**	71.6
traffic-sign	9.1	4.8	3.9	16.7	22.2	18.2	**29.1**
pole	1.3	2.8	6.9	30.1	28.6	31.5	**37.8**
trashcan	0.4	7.4	24.1	28.9	16.3	**48.1**	46.3
building	37.1	57.5	66.1	75.1	73.1	76.3	**79.9**
cone/stone	0.2	0.6	6.6	**58.6**	34.0	27.7	34.5
fence	8.4	12.0	23.4	32.2	40.9	**47.7**	47.3
bike	18.5	35.3	28.6	44.9	50.3	51.4	**53.9**
ground	72.1	71.3	73.5	76.3	79.1	**80.3**	78.4
mean-IoU	12.9	26.9	30.9	43.2	46.4	50.3	**51.9**

Table 7.2: Evaluation results on the SemanticPOSS test set.

Lite-HDseg (*Razani et al.*, 2021), SequeezeSegv3 (*Xu et al.*, 2020a), CENet (*Cheng et al.*, 2022), SalsaNext (*Cortinhal et al.*, 2020b), and MiNet (*Li et al.*, 2021a). The Nearest Label Assignment (NLA) post-processing is used by FIDNet (*Zhao et al.*, 2021). It iterates over each point to check if a point is occluded or not. We use the source code from the corresponding methods. For the Point Refine module proposed in MotionSeg3D (*Sun et al.*, 2022a), we follow its implementation. We use SPVCNN (*Liu et al.*, 2019c) as the Point Refine module and use the features before the classification layer as the input to the Point Refine module. We then fine-tune the network with the Point Refine module in a second stage with a 0.001 learning rate for ten epochs. The results show the "many-to-one" issue harms the performance heavily. Without our proposed post-processing ('w/o MVP'), the mean IoU is by 6.1 lower. That CRF can actually decrease the mean IoU has been also shown in *Milioto et al.* (2019c). While NLA and k-NN improve the results, the best mean IoU is achieved by our approach.

Effect of different fusion strategy. In Table 7.4, we replace the proposed temporal fusion layer with other strategies. Mars3D (*Liu et al.*, 2023a) adopts element-wise summation to aggregate temporal multi-scan point cloud embeddings and produce enhanced features. The temporal memory attention (TMA) module (*Wang et al.*, 2021) validates its effectiveness on the video semantic segmentation task. BEVFormer v2 (*Yang et al.*, 2023) uses a feature warp and concatenation strategy to incorporate temporal information and shows its effectiveness on the LiDAR detection task. We follow its implementation, which concatenates previous BEV features with the current BEV feature along the channel dimension and employs residual blocks for dimensionality reduction. We transform the scans to the same ego-car coordinates to implement the accurate alignment between temporal scans.

	mean-IoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
w/o MVP	60.4	85.8	44.0	61.5	80.3	53.0	68.7	70.2	0.91	**94.8**	42.1	80.9	0.95	81.8	52.4	83.2	60.3	70.6	51.9	47.9
CRF (*Wu et al.*, 2018a)	58.2 (-2.2)	87.0	40.0	57.3	67.7	52.2	66.1	62.5	0.38	94.5	**46.4**	**81.1**	0.66	81.7	53.6	81.4	60.9	66.3	49.0	47.6
PointRefine (*Sun et al.*, 2022a)	59.2 (-1.2)	84.5	43.7	53.7	76.3	48.6	68.3	70.6	7.5	94.6	39.8	80.5	**11.8**	81.4	50.7	83.8	59.4	72.2	51.1	46.1
NLA (*Zhao et al.*, 2021)	64.4 (+4.0)	92.0	47.5	66.8	79.0	55.9	76.2	85.7	**12.4**	94.5	42.7	80.8	10.6	87.3	54.6	85.9	66.0	72.2	63.4	49.8
k-NN (*Milioto et al.*, 2019c)	64.5 (+4.1)	91.4	50.7	66.9	81.2	54.9	76.8	85.1	0.96	94.5	41.6	80.9	0.95	88.5	55.6	86.2	66.8	71.5	64.5	50.2
MVP (Ours)	**66.5** (+6.1)	**93.4**	**54.1**	**70.2**	**85.9**	**59.8**	**79.8**	**88.0**	0.58	94.7	44.8	**81.1**	0.46	**90.3**	**66.6**	**86.8**	**69.5**	**72.7**	**65.1**	**50.3**

Table 7.3: Comparison with different post-processing methods. Our MVP method is significantly better.

Fusion Strategies	mIoU
w/o TCA	66.9
TMA module (*Wang et al.*, 2021)	67.8 (+0.9)
Residual images (*Wang et al.*, 2022b)	61.4 (-5.5)
Element-wise addition (*Liu et al.*, 2023a)	67.6 (+0.7)
Channel concatenation (*Yang et al.*, 2023)	68.0 (+1.1)
TCA module (ours)	**68.1** (+1.2)

Table 7.4: Comparison with other temporal fusion methods.

For the LiDAR semantic segmentation task, Meta-RangeSeg (*Wang et al.*, 2022b) proposes to use three previous residual images as input and a meta-kernel module to incorporate temporal information. We follow its implementation and add to the five-channel input (x,y,z,r,i) three channels for the three residual images and a channel for the mask, which indicates whether the pixel is a projected 3D point or not. The residual images are calculated by first transforming the point clouds of previous frames into the coordinates of the current frame and then calculating the absolute differences between the range values of the current scan and the transformed one with normalization. A meta-kernel is followed to capture the spatial and temporal information. For a fair comparison, we keep the encoder and decoder of our architecture. We report the projection-based mIoU here because the loss function is applied directly to the range image. All strategies are trained with the same setting and pipeline. The results in Table 7.4 show that our temporal fusion approach performs best.

7.3.3 Generalization Ability

In order to demonstrate that the proposed max-voting-based post-processing (MVP) can be applied to other networks, we apply MVP to various networks including point-based methods (*Choy et al.*, 2019; *Liu et al.*, 2019c), projection-based methods (*Wang et al.*, 2022b; *Cheng et al.*, 2022; *Zhao et al.*, 2021), and a voxel-based method (*Zhang et al.*, 2020a). In contrast to Table 7.1, which contains the reported results on the test set, Table 7.5 reports the results of the public available pre-trained models with and without post-processing on the validation set. The results show that the proposed MVP post-processing improves the results for various approaches.

Backbone	Input	Post-processing		
		-	k-NN (*Milioto et al.*, 2019c)	MVP (Ours)
MinkUNet (*Choy et al.*, 2019)	Point	58.9	/	**60.1** (+1.1)
SPVNAS (*Liu et al.*, 2019c)		64.7	/	**65.3** (+0.6)
FIDNet (*Zhao et al.*, 2021)	Range	55.4	58.6 (+3.2)	**61.5** (+6.1)
Meta-RangeSeg (*Wang et al.*, 2022b)		56.6	60.3 (+3.7)	**63.1** (+6.5)
CENet (*Cheng et al.*, 2022)		58.8	62.6 (+3.8)	**64.7** (+5.9)
PolarSeg (*Zhang et al.*, 2020a)	Polar	57.2	/	**58.0** (+0.8)

Table 7.5: Generalization to other networks.

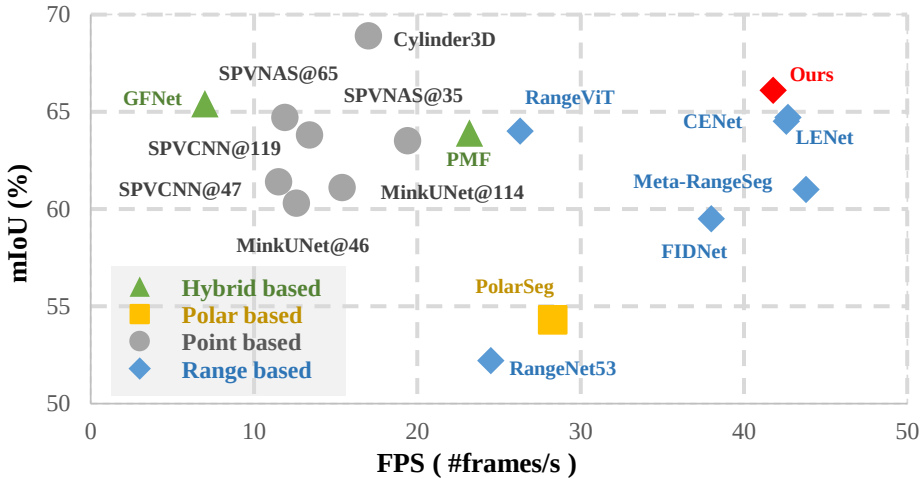


Figure 7.4: mIoU vs. runtime on SemanticKITTI. Our method balances mIoU and inference time better than other state-of-the-art methods. Best viewed in color.

7.3.4 Impact of other factors

Effect of frame numbers. In Figure 7.6 (a), we analyze the impact of the length L of the sliding window update discussed in Section 7.2.3. It shows that $L = 10$ is a reasonable choice.

Effect of grid size resolution. As discussed in Section 7.2.3, we quantize the aggregated scans into a voxel grid with a fixed resolution. Note that the resolution needs to be appropriate since the underlying assumption is that all points inside a voxel share the same semantic class. This assumption is violated if the voxel size is too large and the estimates become noisy when it is too small. We evaluate the impact of different resolutions in Figure 7.6 (b). The results show that a voxel size of 0.10m works best.

Qualitative evaluation. We provide a visual comparison of two point clouds in Figure 7.5. The impact of the "many-to-one" problem is evident when there is no post-processing technique applied, shown in Figure 7.5 (a). Specifically, points of the trunk of the tree inherit the predictions from points of the traffic sign and vegetation since they are projected to the same range pixel. In Figure 7.5 (b), the widely-used k-NN (*Milioto et al.*, 2019c) refines the predictions, but it cannot correct false predictions when larger areas are occluded. In contrast, our method corrects false predictions in the single scan by keeping a consistent prediction across previous scans, as shown in Figure 7.5 (c). This demonstrates the benefit of introducing temporal information in the post-processing step to resolve the "many-to-one" problem.

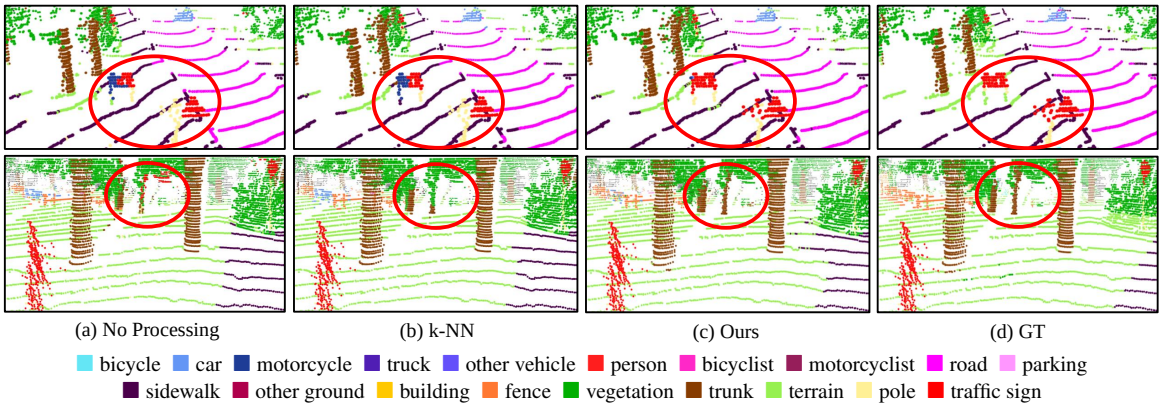
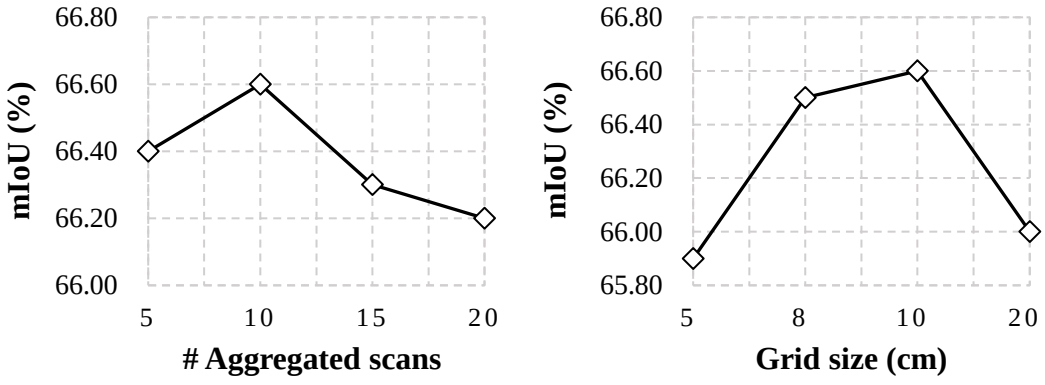


Figure 7.5: Qualitative analysis of the post-processing scheme. (a) The "many-to-one" issue is evident without post-processing, e.g., the trunk is partially segmented as traffic sign and vegetation as they project onto the same range pixel (row 2). (b) k-NN (*Milioto et al.*, 2019c) smooths the semantic labels locally, but it cannot resolve ambiguities by objects that are close or prediction errors. (c) Our method exploits temporal information to resolve false predictions (row 1) or ambiguities due to occlusions (row 2). Best viewed in color.



(a) Effect of number of scans. (b) Effect of grid size.

Figure 7.6: Effect of window size and grid size resolution.

7.4 Conclusion

In this chapter, we introduced a novel solution named TFNet to tackle the issue of boundary blurriness, also called "many-to-one", in projection-based LiDAR semantic segmentation tasks. Our approach leverages temporal information by proposing temporal fusion layers during the training process and a sequential max voting strategy during inference. The experiments on two benchmarks demonstrate the advantages of TFNet. In particular, integrating temporal information enables TFNet to exhibit robustness in scenarios with significant occlusion while maintaining real-time performance. Furthermore, we conducted comprehensive ablation studies to validate the design, as well as the generalization of the proposed post-processing to other networks.

For now, we have presented several solutions for efficient LiDAR point cloud semantic segmentation. The core idea behind these methods is the utilization of projection maps of LiDAR scans. We start with an efficient architecture in Chapter 4 which directly operates on the projection map. Then we explore the relationship between point-based methods and projection-based methods for LiDAR point cloud semantic segmentation in Chapter 6. We have figured out the reason for the poor performance and low efficiency of point-based methods for this task and proposed a general strategy to improve the performance. In Chapter 7, we investigate projection-based methods and voxel-based methods and try to find the reason that projection-based methods usually cannot achieve similar performance to voxel-based methods. We found this is usually caused by "many-to-one" issue and we proposed to utilize the temporal information to fix it. The temporal information is also utilized in Chapter 5 where we propose a new data representation to compress moving information. With this data representation, we can achieve moving object segmentation accurately and efficiently.

CHAPTER 8

Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting

Individual Contribution

The following chapter is based on the publication:

Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting
ShiJie Li, Yanying Zhou, Jinhui Yi, Juergen Gall
International Conference on Computer Vision (ICCV), 2021.
doi:10.1109/ICCV48922.2021.00195

Below we will summarize the contributions of each author.

- **Shijie Li**
  Shijie proposed the main idea of the paper and implemented the whole method. Shijie performed the experiments and conducted the qualitative and quantitative analysis. Shijie also wrote most of the paper.
- **Yanying Zhou**
  Yanying contributed to the main idea and writing. Yanying also helped to draw the figures in the paper.
- **Jinhui Yi**
  Jinhui contributes to writing.
- **Juergen Gall**
  Juergen supervised the project. He contributed with discussion and writing.

In the previous chapters, we present some solutions for efficient LiDAR point cloud semantic segmentation. At this stage, the vehicles can understand the nearby environment. Based on this, we go one step further and conduct research on motion forecasting, which is important for ensuring the safety of autonomous driving. In the following, we convert our focus to motion forecasting in this chapter. Trajectory forecasting is a crucial step for autonomous vehicles and mobile robots in order to navigate and interact safely. In this chapter, we mainly focus on mapless human trajectory prediction as the safety of humans is the highest priority. In order to handle the spatial interactions between objects, graph-based approaches have been proposed. These methods, however, model motion on a frame-to-frame basis and do not provide a strong temporal model. To overcome this limitation, we propose a compact model called Spatial-Temporal Consistency Network (STC-Net). In STC-Net,

((a)) Social-STGCNN (*Mohamed et al.*, 2020) ((b)) Ours

Figure 8.1: State-of-the-art graph-based approaches like (*Mohamed et al.*, 2020) do not model the temporal motion of the trajectories well, which results in shaky and unrealistic trajectories (dashed yellow) as shown in (a). In this figure, red denotes the observed trajectory and blue is the ground-truth future trajectory. Our approach addresses this issue and forecasts spatially and temporally consistent trajectories (b).

dilated temporal convolutions are introduced to model long-range dependencies along each trajectory for better temporal modeling while graph convolutions are employed to model the spatial interaction among different trajectories. Furthermore, we propose a feature-wise convolution to generate the predicted trajectories in one pass and refine the forecast trajectories together with the reconstructed observed trajectories. We demonstrate that STC-Net generates spatially and temporally consistent trajectories and outperforms other graph-based methods.

Contents

8.1 Introduction

Intelligent agents like autonomous vehicles and mobile robots navigate and interact in spaces that are shared with humans. The safety of humans is therefore of the highest priority. To this end, agents are required to understand and forecast the trajectories of surrounding pedestrians or vehicles such that they can make the right decisions and for instance navigate safely and smoothly through a crowd. Trajectory forecasting, however, is challenging due to the complex behavior and interactions of humans in crowds. Furthermore, the available resources will always be a limiting factor as the agents are equipped with energy-efficient hardware. Hence, there is a need for compact forecasting models with a very low latency. In practice, low latency is required for most applications since there is otherwise not enough time to adjust the motion and any millisecond can save lives.

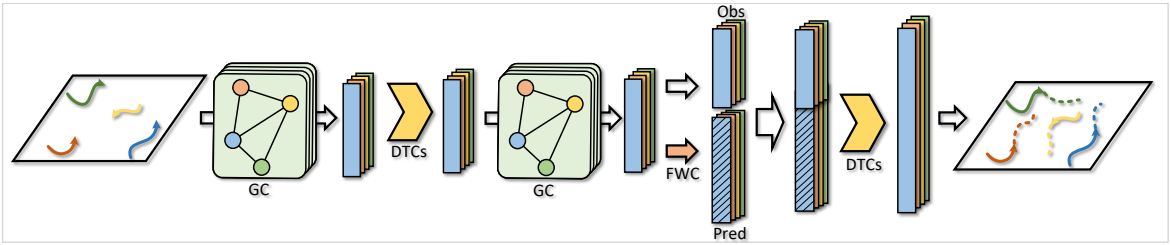


Figure 8.2: The proposed spatial-temporal consistency network takes observed trajectories (solid curves) of N objects as input and forecasts the future trajectories (dashed curves). It combines graph convolutions (GC) for modeling interactions of trajectories in close proximity and dilated temporal convolutions (DTCs) for capturing temporal relations over the entire trajectories. The feature-wise convolution (FWC) forecasts the features in one pass and the final refinement ensures the consistency of the reconstructed and forecast part of a trajectory.

Over the last few years, several approaches have been proposed for trajectory forecasting. RNN-based methods (*Alahi et al.*, 2016; *Liang et al.*, 2019; *Zhang et al.*, 2019b) utilize a recurrent model to model the trajectory of each pedestrian in the scene, and their interactions are taken into account by a pooling operation. GAN-based methods (*Gupta et al.*, 2018; *Li et al.*, 2019; *Sadeghian et al.*, 2019) enable RNN-based methods to produce multiple socially plausible trajectories. Graph-based methods (*Kosaraju et al.*, 2019; *Williams and Li*, 2018; *Mohamed et al.*, 2020) model spatial relations as a graph and utilize graph convolutions. These methods, however, have in common that they use recurrent neural networks to generate future trajectories, which results in a relatively high latency. To address this issue, Social-STGCNN (*Mohamed et al.*, 2020) proposed a CNN for time-extrapolation. As a result, Social-STGCNN achieves a very low latency of 2ms. This, however, comes at the cost that the forecast trajectories are noisy and not temporally consistent as shown in Figure 8.1(a).

In this work, we therefore address this issue and present an approach that generates spatially and temporally consistent trajectories with a latency of less than 2ms. To achieve this, the proposed Spatial-Temporal Consistency Network (STC-Net), which is shown in Figure 8.2, utilizes graph convolutions for spatial modeling and dilated temporal convolutions for temporal modeling. Our network generates trajectories that are consistent over the past and the future. This consistency is implicitly enforced by reconstructing the past, jointly refining the reconstructed past and forecast future, and computing the loss over the entire trajectory. In order to achieve a very low latency, we do not use any recurrent layers but propose feature-wise convolutions that generate the future trajectories with variable length in one pass. The proposed network design yields very compact networks with only 0.7k parameters and 1.3ms latency.

We thoroughly evaluate the approach on three datasets with six different scenes and analyze its generalization ability by performing a cross-dataset experiment and evaluating the approach for different temporal sampling rates. The proposed approach outperforms other graph-based methods in terms of accuracy, model size, and latency. It is also important to note that most approaches cannot keep up with the frame rate, i.e., the forecasting cannot be performed until the next frame is captured. This means that these methods are not fast enough for real-world applications. Even for a very slow frame rate of 2.5 frames per second, only *Gupta et al.* (2018); *Monti et al.* (2020); *Mohamed et al.* (2020) are able to process the data fast enough. Compared to these methods, our approach is not only faster, but it is also far more accurate.

8.2 Method

The goal of trajectory forecasting is to predict the upcoming trajectories given past observed trajectories. We denote the observed trajectory of each object n, which are commonly pedestrians but can also be other objects like vehicles, by $\mathcal{T}_o^n = \{\mathbf{p}_t^n = (x_t^n, y_t^n) \mid t \in \{1, ..., T_o\}\}$ and the future trajectory by $\mathcal{T}_f^n = \{\mathbf{p}_t^n = (x_t^n, y_t^n) \mid t \in \{T_o + 1, ..., T_o + T_f\}\}$, where $\mathbf{p}_t^n$ is the location of the object n at time t.

For forecasting, we use a Gaussian distribution $\mathcal{N}_{\mathbf{p}_t^n}(\hat{\mu}_t^n, \hat{\Sigma}_t^n)$, *i.e.*we estimate for each $\mathbf{p}_t^n \in \mathcal{T}_f^n$ the mean $\hat{\mu}_t^n$ and the covariance matrix $\hat{\Sigma}_t^n$. In order to vectorize the covariance matrix, we use

$$\hat{\Sigma} = \begin{pmatrix} \hat{\sigma}_1^2 & \hat{\rho}\hat{\sigma}_1\hat{\sigma}_2 \\ \hat{\rho}\hat{\sigma}_1\hat{\sigma}_2 & \hat{\sigma}_2^2 \end{pmatrix}. \tag{8.1}$$

The covariance matrix is then defined by the three-dimensional vector $(\hat{\sigma}_1, \hat{\sigma}_2, \hat{\rho})$.

For training, we minimize the negative log-likelihood, *i.e.*given for each trajectory the ground-truth position $\mathbf{p}_t^n$, we minimize

$$L_f = -\sum_{n=1}^{N} \sum_{t=T_o+1}^{T_o+T_f} \log\left(\mathcal{N}_{\mathbf{p}_t^n}(\hat{\mu}_t^n, \hat{\Sigma}_t^n)\right), \tag{8.2}$$

where N is the number of present objects.

In contrast to previous works, our proposed network not only estimates the future trajectory $\mathcal{T}_f^n$, but also reconstructs the observed trajectory $\mathcal{T}_o^n$ and refines them both as a single trajectory as it is illustrated in Figure 8.2. This helps to learn a better representation that is spatially and temporally consistent as we will show in the experiments. For the observed trajectory $\mathcal{T}_o^n$, we use the squared error as reconstruction loss:

$$L_o = \sum_{n=1}^{N} \sum_{t=1}^{T_o} \|\hat{\mathbf{p}}_t^n - \mathbf{p}_t^n\|^2, \tag{8.3}$$

where $\hat{\mathbf{p}}_t^n$ is the reconstructed position for object n at frame t and $\mathbf{p}_t^n$ is the observed position.

For training the network, we sum the two loss functions Equation 8.2 and Equation 8.3 with equal weights, *i.e.*

$$L = L_f + wL_o. \tag{8.4}$$

where $w = 1$. For the ablation study, however, we also investigate different values of w.

8.2.1 Spatial Graph Representation

As in other graph-based approaches (*Kosaraju et al.*, 2019; *Williams and Li*, 2018; *Mohamed et al.*, 2020), we use a graph structure to model the spatial relations between the objects. Since we focus on the temporal modeling, we use for a fair comparison the graph representation that has been proposed in *Mohamed et al.* (2020). At each timestamp t, a spatial graph $G_t = (V_t, E_t)$ is constructed where $V_t = \{v_t^n \mid n \in \{1, ..., N\}\}$ are the vertices corresponding to the N objects and $E_t = \{e_t^{nm} \mid n, m \in \{1, ..., N\}\}$ denote the edges that encode the spatial relationship between object n and m. For each edge, the affinity a_t^{nm} is computed by

$$a_t^{nm} = \begin{cases} \|\mathbf{p}_t^n - \mathbf{p}_t^m\|^{-1} & \text{, if} \quad \|\mathbf{p}_t^n - \mathbf{p}_t^m\| > 0 \\ 0 & \text{, otherwise.} \end{cases} \tag{8.5}$$

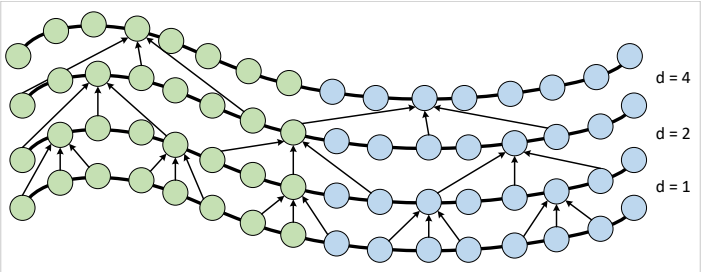


Figure 8.3: We model the observed trajectories (green) and forecast trajectories (blue) together by using dilated temporal convolutions that are stacked with increasing dilation rates. This provides a very strong temporal model for trajectories since the motion is modeled over a large temporal receptive field.

This means that the objects influence each other when they are close. Suppose $V^{(l)}$ is the stack of $V_t^{(l)}$, that is the vertices at time step t and network layer l. The features of the vertices $f(V^{(l)})$ are updated in a graph convolution layer as follows:

$$f\left(V^{(l+1)}\right) = ReLU\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} f\left(V^{(l)}\right) W^{(l)}\right) \tag{8.6}$$

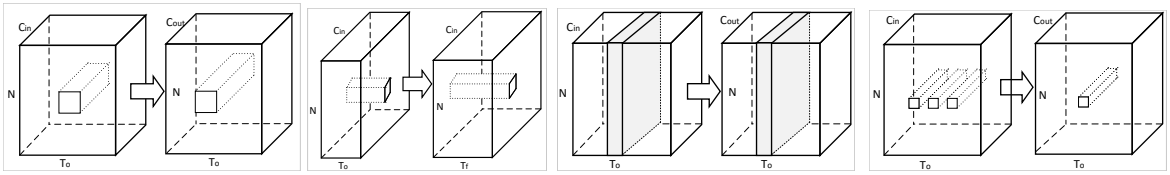
where $W^{(l)}$ are the learnable convolution weights, $\tilde{A}$ is the stack of $\tilde{A}_t = A_t + I$, and the diagonal matrix $\tilde{D}$ is the stack of $\tilde{d}_t^{nn} = \sum_m \tilde{a}_t^{nm}$.

While Equation 8.6 describes the layer for a single frame t, some works simply stack the features and matrices for multiple frames in order to extend the graph to the temporal domain. This, however, results in a poor temporal model that generates noisy trajectories that are not very consistent over time as we show in the experiments. In this work, we, therefore, propose to use dilated temporal and feature-wise convolutions for forecasting trajectories and combine them with graph convolutional layers. In the experiments, we will show that this results in forecast trajectories that are spatially and temporally more consistent.

8.2.2 Architecture

In order to forecast spatially and temporally consistent trajectories, we propose to combine different convolutions as shown in Figure 8.4. While graph convolutions model the relations of objects that are close, they are not the best choice for temporal modeling as discussed previously. We therefore propose graph convolutions for spatial modeling and dilated temporal convolutions for temporal modeling. In contrast to graph convolutions that only consider neighbors based on the affinity matrix A, dilated temporal convolutions with gradually increasing dilation rates capture long-range dependencies as it is illustrated in Figure 8.3.

For our network, which is shown in Figure 8.2, we first aggregate spatial information among different observed trajectories at each frame using graph convolutions. We then aggregate temporal information using dilated temporal convolutions (*Li et al.*, 2020a). We use 3 dilated temporal convolutions with kernel size 3 and dilation rates 2^{l-1}, where l is the layer number, and apply it to each trajectory. Finally, we use a graph convolution to aggregate again the spatial information. In this way, the network is able to consider the interaction of trajectories that are in close proximity as well as the full temporal information that is available for each trajectory.



((a)) Vanilla 1×1 Convolution ((b)) Feature-wise Convolution ((c)) Graph Convolution ((d)) Dilated Temporal Convolution

Figure 8.4: The four different convolutions that are used in STC-Net. For the illustrations, the input tensor is (C_{in}, T_o, N) where T_o are the observed frames, N denotes the number of objects, and C_{in} is the dimensionality of the input features. (a) Vanilla 1×1 convolutions map the feature to another space with dimensionality C_{out}. The output tensor is thus (C_{out}, T_o, N). (b) In contrast, the proposed feature-wise convolutions keep the feature dimensionality but adjust the temporal dimension. Since the predicted length T_f is variable, it is suitable for forecasting and we use them to forecast the future features. The output tensor is (C_{in}, T_f, N). (c) Graph convolutions aggregate the features at a frame based on the spatial relations of the objects and they are used to model interactions between objects. Depending on the number of kernels, the feature dimensionality can change and the output tensor is (C_{out}, T_o, N). (d) Temporal convolutions operate over time and aggregate for each trajectory temporal information. Due to dilated convolution kernels, the temporal convolutions can take the entire trajectory into account. While graph convolutions model spatial relations, temporal convolutions model temporal relations.

After the interaction of both spatial and temporal information, the initial trajectory prediction is generated by a feature-wise convolution, which is illustrated in Figure 8.4. It is similar to a 1×1 convolution, but the main difference is the information aggregation dimension. At this stage of the network, the size of the feature tensor is (C_{in}, T_o, N) where C_{in} is the feature dimension, T_o is the length of the observed trajectories, and N is the number of objects in the current scene. While a vanilla 1×1 convolution changes the feature dimensionality and aggregates the information across the features, a feature-wise convolution aggregates for each channel in the feature vector all observed information along the temporal dimension and can be used to predict the future for each channel with variable length. After the feature-wise convolution, the size of the feature tensor is (C_{in}, T_f, N) where T_f is the length of the future trajectories. In contrast to recurrent or auto-regressive approaches, feature-wise convolution predicts the future in one pass.

Different from previous works that only predict future trajectories, we also reconstruct the observed trajectories to ensure that the observed and forecast parts of a trajectory are consistent. As shown in Figure 8.2, we refine both parts as a single consistent trajectory. In particular, the features of the observed and forecast trajectories are concatenated and refined by dilated temporal convolutions as before. The only difference is that we use 5 layers since the concatenated trajectory is longer than the observed one. For training, we use the loss Equation 8.4.

	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
S-LSTM (*Alahi et al.*, 2016)	1.09 / 2.35	0.79 / 1.76	0.67 / 1.40	0.47 / 1.00	0.56 / 1.17	0.72 / 1.54
PIF (*Liang et al.*, 2019)	0.73 / 1.65	0.30 / 0.59	0.60 / 1.27	0.38 / 0.81	0.31 / 0.68	0.46 / 1.00
SR-LSTM (*Zhang et al.*, 2019b)	0.63 / 1.25	0.37 / 0.74	0.51 / 1.10	0.41 / 0.90	0.32 / 0.70	0.45 / 0.94
DS-LSTM (*Chaofan Tao and Luo*, 2020)	0.66 / 1.21	0.27 / 0.46	0.50 / 1.07	0.33 / 0.68	0.28 / 0.60	0.41 / 0.80
FvTraj (*Bi et al.*, 2020)	0.56 / 1.14	0.28 / 0.55	0.52 / 1.12	0.37 / 0.78	0.32 / 0.68	0.41 / 0.85
S-GAN (*Gupta et al.*, 2018)	0.81 / 1.52	0.72 / 1.61	0.60 / 1.26	0.34 / 0.69	0.42 / 0.84	0.58 / 1.18
SoPhie (*Sadeghian et al.*, 2019)	0.70 / 1.43	0.76 / 1.67	0.54 / 1.24	0.30 / 0.63	0.38 / 0.78	0.54 / 1.15
CGNS (*Li et al.*, 2019)	0.62 / 1.40	0.70 / 0.93	0.48 / 1.22	0.32 / 0.59	0.35 / 0.71	0.49 / 0.97
Social-Ways (*Amirian et al.*, 2019)	0.39 / 0.64	0.39 / 0.66	0.55 / 1.31	0.44 / 0.64	0.51 / 0.92	0.46 / 0.83
Goal-GAN (*Dendorfer et al.*, 2020)	0.59 / 1.18	0.19 / 0.35	0.60 / 1.19	0.43 / 0.87	0.32 / 0.65	0.43 / 0.85
TPNet (*Fang et al.*, 2020)	0.84 / 1.73	0.24 / 0.46	0.42 / 0.94	0.33 / 0.75	0.26 / 0.60	0.42 / 0.90
AgentFormer (*Yuan et al.*, 2021)	0.31 / 0.18	0.16 / 0.09	0.35 / 0.15	0.37 / 0.18	0.73 / 0.21	0.38 / 0.16
PECNet (*Mangalam et al.*, 2020)	0.54 / 0.87	0.18 / 0.24	0.35 / 0.60	0.22 / 0.39	0.17 / 0.30	0.29 / 0.48
MemoNet (*Xu et al.*, 2022)	0.13 / 0.07	0.09 / 0.06	0.19 / 0.13	0.20 / 0.11	0.40 / 0.16	0.20 / 0.11
MID (*Gu et al.*, 2022)	0.39 / 0.66	0.13 / 0.22	0.22 / 0.45	0.17 / 0.30	0.13 / 0.27	0.21 / 0.38
Leapfrog (*Mao et al.*, 2023)	0.39 / 0.58	0.11 / 0.17	0.26 / 0.43	0.18 / 0.26	0.13 / 0.22	0.21 / 0.33
STSGN (*Zhang et al.*, 2019a)	0.75 / 1.63	0.63 / 1.01	0.48 / 1.08	0.30 / 0.65	0.26 / 0.57	0.48 / 0.99
Social-BiGAT (*Kosaraju et al.*, 2019)	0.69 / 1.29	0.49 / 1.01	0.55 / 1.32	0.30 / 0.62	0.36 / 0.75	0.48 / 1.00
RSBG (*Sun et al.*, 2020)	0.80 / 1.53	0.33 / 0.64	0.59 / 1.25	0.40 / 0.86	0.30 / 0.65	0.48 / 0.99
Social-STGCNN (*Mohamed et al.*, 2020)	0.64 / 1.11	0.49 / 0.85	0.44 / 0.79	0.34 / 0.53	0.30 / 0.48	0.44 / 0.75
STGAT (*Huang et al.*, 2019)	0.65 / 1.12	0.35 / 0.66	0.52 / 1.10	0.34 / 0.69	0.29 / 0.60	0.43 / 0.83
STC-Net	0.66 / 1.28	0.33 / 0.58	0.41 / 0.78	0.29 / 0.51	0.27 / 0.45	0.39 / 0.72
STC-Net-NAS	0.64 / 1.18	0.33 / 0.54	0.39 / 0.74	0.29 / 0.49	0.26 / 0.45	0.38 / 0.68

Table 8.1: Results on the ETH (*Pellegrini et al.*, 2009) and UCY (*Lerner et al.*, 2007) datasets.

8.3 Experiment

8.3.1 Datasets

We evaluate our method on three common trajectory forecasting datasets which are described below. As in previous works, 8 frames are used as observation and the following 12 frames need to be forecast. All trajectories are provided in 2D coordinates.

The **ETH** (*Pellegrini et al.*, 2009) and **UCY** (*Lerner et al.*, 2007) datasets aim at pedestrian trajectory forecasting. ETH contains 2 top view scenes with 750 trajectories and UCY contains 3 scenes close to the top view with 786 trajectories. The data processing is done as *Gupta et al.* (2018). The sampling rate is 2.5 frames per second (one frame per 0.4s), which means that the model observes 3.2 seconds and predicts the trajectories for the next 4.8 seconds.

The **Stanford Drone Dataset** (*Robicquet et al.*, 2016) includes a series of videos captured by a hovering drone from 8 different college campuses with more than 10,000 trajectories. The dataset is much larger than ETH and UCY and includes other objects like bikes or cars apart from pedestrians. The frame rate is 2.5 frames per second, which is the same as for ETH and UCY. We evaluate our approach as *Monti et al.* (2020) for two protocols, namely 2D world and 2D image coordinates.

8.3.2 Metrics

There are two commonly used metrics to evaluate the performance of trajectory forecasting methods: average displacement error (ADE) and final displacement error (FDE) (*Pellegrini et al.*, 2009).

	ADE	FDE
Social LSTM$^+$ (*Alahi et al.*, 2016)	0.73	0.96
Social-STGCNN$^+$ (*Mohamed et al.*, 2020)	0.71	1.14
Social-Ways (*Amirian et al.*, 2019)	0.62	1.16
STGAT (*Huang et al.*, 2019)	0.58	1.11
STGAT$^+$ (*Huang et al.*, 2019)	0.63	1.19
DAG-Net (*Monti et al.*, 2020)	0.54	1.05
DAG-Net* (*Monti et al.*, 2020)	0.53	1.02
STC-Net	0.51	0.85
STC-Net-NAS	**0.49**	**0.82**

Table 8.2: Results on the Stanford Drone dataset (world coordinates) (*Robicquet et al.*, 2016). $^+$ means that we train the model with the code offered by the authors. * denotes that we use the pre-trained model offered by the authors.

$\mathcal{T}_o$	NLL	NLL	L2	L2
$\mathcal{T}_f$	NLL	L2	NLL	L2
ADE / FDE	0.63 / 1.09	1.00 / 1.95	**0.51 / 0.85**	0.93 / 1.79

Table 8.3: Impact of different loss terms for the reconstructed ($\mathcal{T}_o$) and forecast ($\mathcal{T}_f$) trajectory. While NLL denotes the negative log-likelihood in Equation 8.2, L2 denotes the L2 loss in Equation 8.3.

The average displacement error is defined as:

$$ADE = \frac{\sum_{n=1}^{N}\sum_{t=T_o+1}^{T_o+T_f} \|\hat{\mathbf{p}}_\mathbf{t}^\mathbf{n} - \mathbf{p}_\mathbf{t}^\mathbf{n}\|_2}{N \times T_f}, \tag{8.7}$$

where $\hat{\mathbf{p}}_\mathbf{t}^\mathbf{n}$ is the estimated position of object n at frame t and $\mathbf{p}_\mathbf{t}^\mathbf{n}$ is the ground-truth position. It measures the average prediction performance along the whole trajectory. The final displacement error (FDE) is defined as:

$$FDE = \frac{\sum_{n=1}^{N} \left\|\hat{\mathbf{p}}_{\mathbf{T_o}+\mathbf{T_f}}^\mathbf{n} - \mathbf{p}_{\mathbf{T_o}+\mathbf{T_f}}^\mathbf{n}\right\|_2}{N}. \tag{8.8}$$

In contrast to ADE, it only measures the prediction accuracy at the endpoint. Because the prediction of our method is a distribution, we follow the evaluation protocol used in Social-LSTM (*Alahi et al.*, 2016) to compare the predicted distribution with the ground truth value. Specifically, we sample 20 predictions from the predicted distribution and take the closest sample to the ground truth to compute ADE and FDE.

8.3.3 Ablation Study

We conduct the ablation study on the Stanford Drone Dataset (*Robicquet et al.*, 2016) since it is the largest dataset and has the highest diversity. We first evaluate the proposed loss function Equation 8.4, which uses the L2 loss for reconstructed trajectory in Equation 8.3 and negative log-likelihood (NLL) for the forecast trajectory in Equation 8.2. The results are shown in Table 8.3.

From the table we can see that applying the L2 loss on the reconstructed observed trajectory and the negative log-likelihood on the forecast trajectory achieves the lowest error (col 3). This is reasonable because the observed trajectory is determined, which means that there is no uncertainty in

w	0.5	0.7	1.0	1.3	w/o obs	w/o refine
ADE / FDE	0.55 / 0.91	0.54 / 0.87	**0.51 / 0.85**	0.54 / 0.91	0.58 / 0.97	0.60 / 1.04

Table 8.4: Impact of w in Equation 8.4. While 'w/o obs' denotes a setting where the reconstructed observed trajectories are not concatenated with the forecast trajectories, 'w/o refine' uses the forecast trajectories directly after the feature-wise convolution without the last layer of dilated temporal convolutions.

	ADE	FDE
SoPhie (*Sadeghian et al.*, 2019)	16.27	29.38
Social GAN (*Gupta et al.*, 2018)	27.23	41.44
CF-VAE (*Bhattacharyya et al.*, 2019)	12.6	22.3
P2TIRL (*Deo and Trivedi*, 2020)	12.58	22.07
SimAug$^+$ (*Liang et al.*, 2020a)	10.27	19.71
PECNet (*Mangalam et al.*, 2020)	**9.96**	**15.88**
STC-Net	11.96	20.12
STC-Net-NAS	11.88	20.08

Table 8.5: Results on the Stanford Drone dataset (image coordinates) (*Robicquet et al.*, 2016). $^+$ the approach uses additional training data.

the ground truth. In contrast, the future trajectory can be uncertain since the future is not necessarily determined and several trajectories might be plausible. If we use the L2 loss for the forecast trajectory (col 2 and 4), the error drastically increases since the L2 loss does not handle the uncertainty of the future. On the other hand, if we use only the negative log-likelihood (col 1), the error is higher than for the proposed setting (col 3). This is expected since the L2 error enforces a more accurate reconstruction of the observed trajectory, which is beneficial for the forecasting task.

As discussed previously, we do not weigh the two loss terms in Equation 8.4. Nevertheless, we evaluate in Table 8.4 the impact of an additional weighting parameter w. We can see that the error first decreases as w increases to 1.0 and then increases again when w is larger than 1.0. This shows that there is no need to weight the two loss functions. In the second to last column, we show why it is important to take the observed trajectory for the final estimate into account. If we do not concatenate the reconstructed observed and the forecast trajectories (w/o obs) but refine only the forecast trajectory, the error increases. If we do not refine the trajectories at all (w/o refine), the error even increases further. This shows the importance of the additional refinement layers.

In Table 8.9, we analyze the impact of the used convolutions on the latency as well as accuracy. Since we cannot simply remove the convolutions, we replaced them by other operations. We used pooling instead of graph convolutions (w/o GC), temporal convolutions instead of dilated temporal convolutions (w/o DTC), and the TXP-CNN of Social-STGCNN (*Mohamed et al.*, 2020) instead of the feature-wise convolution (w/o FWC). The results show that both graph convolutions and dilated temporal convolutions are required in order to achieve accurate predictions. It also shows that the graph convolutions take more than 50% of the processing time. Furthermore, the proposed feature-wise convolution is much more efficient and effective compared to TXP-CNN.

	0.4s	0.8s	1.2s	1.6s	2.0s	AVG
Social-STGCNN (*Mohamed et al.*, 2020)	0.71 / 1.15	0.73 / 1.14	0.69 / 1.10	0.61 / 0.92	0.67 / 1.01	0.68 / 1.06
DAG-Net (*Monti et al.*, 2020)	0.53 / 1.02	0.53 / 1.01	0.58 / 1.14	0.56 / 1.11	0.82 / 1.64	0.60 / 1.18
STGAT (*Huang et al.*, 2019)	0.63 / 1.19	0.64 / 1.17	0.61 / 1.14	0.51 / 0.92	0.57 / 1.05	0.59 / 1.09
STC-Net	0.51 / 0.85	0.50 / **0.80**	0.50 / 0.83	0.43 / 0.69	0.50 / 0.80	0.49 / 0.79
STC-Net-NAS	**0.49 / 0.82**	**0.49 / 0.80**	**0.48 / 0.82**	**0.41 / 0.67**	**0.47 / 0.78**	**0.47 / 0.78**

Table 8.6: Temporal robustness. The errors (ADE / FDE) for different sampling rates on the Stanford Drone dataset (*Robicquet et al.*, 2016).

	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
Social-STGCNN (*Mohamed et al.*, 2020)	0.64 / **0.88**	0.44 / 0.67	0.50 / 0.79	0.53 / 0.71	0.48 / 0.70	0.52 / 0.75
STGAT (*Huang et al.*, 2019)	0.71 / 1.31	0.33 / 0.61	0.61 / 1.26	0.43 / 0.83	0.38 / 0.75	0.49 / 0.95
DAG-Net (*Monti et al.*, 2020)	0.71 / 1.23	0.22 / 0.41	0.70 / 1.47	0.43 / 0.88	0.29 / 0.60	0.47 / 0.92
STC-Net	0.61 / 1.10	**0.20** / 0.27	0.43 / 0.79	0.35 / 0.59	0.28 / 0.47	0.37 / 0.64
STC-Net-NAS	**0.59** / 1.12	**0.20 / 0.25**	**0.40 / 0.75**	**0.33 / 0.55**	**0.27 / 0.46**	**0.36 / 0.63**

Table 8.7: Cross-scene robustness. For this experiment, the approaches are trained on the Stanford Drone dataset (*Robicquet et al.*, 2016) and evaluated on the ETH (*Pellegrini et al.*, 2009) and UCY (*Lerner et al.*, 2007) datasets.

8.3.4 Comparison with state-of-the-art methods

Accuracy. We compare our method with other state-of-the-art methods on the ETH (*Pellegrini et al.*, 2009), UCY (*Lerner et al.*, 2007), and Stanford Drone (*Robicquet et al.*, 2016) datasets. The results for the ETH (*Pellegrini et al.*, 2009) and UCY (*Lerner et al.*, 2007) datasets are shown in Table 8.1. When we compare our approach to other graph-based methods (*Zhang et al.*, 2019a; *Kosaraju et al.*, 2019; *Sun et al.*, 2020; *Mohamed et al.*, 2020; *Huang et al.*, 2019), we observe that our approach achieves the lowest average (ADE) and final displacement error (FDE) among graph-based methods. When we compare our approach to the state-of-the-art, we observe that only PECNet (*Mangalam et al.*, 2020) achieves a lower error. PECNet uses a two-step approach that first predicts the end-points and then the trajectories based on the predicted endpoints. While predicting end-points is complementary to our approach, the high accuracy comes at a high computational cost as shown in Table 8.8. PECNet requires 3,000 times more parameters and is 467 times slower. In fact, a latency of over 600ms is too high for applications. With recent progress of the diffusion model, MID (*Gu et al.*, 2022) and Leapfrog (*Mao et al.*, 2023) also achieve good performance. However, they need to recursively produce predictions thus inefficient.

As discussed previously, we also generated an instance of the spatial-temporal consistency networks by network architecture search. It needs to be noted that we used only the Stanford Drone Dataset (*Robicquet et al.*, 2016) for the network optimization since it is the largest dataset. If we use this instance, which is denoted by STC-Net-NAS, for the other two datasets, we observe that the error is slightly reduced.

The results for the Stanford Drone dataset (*Robicquet et al.*, 2016) are reported in Table 8.2 and Table 8.5, respectively. The results confirm the very good results from ETH and UCY. Our approach achieves a much lower error compared to other graph-based approaches. Only the recent approaches (*Liang et al.*, 2020a; *Mangalam et al.*, 2020) achieve a lower accuracy. While PECNet (*Mangalam et al.*, 2020) uses a much bigger network and is much slower than our approach as discussed before,

	Parameters	Inference time
PECNet (*Mangalam et al.*, 2020)	2.10M (3000x)	0.6070s (467x)
Social LSTM (*Alahi et al.*, 2016)	264K (377.1x)	1.1800s (907.7x)
SR-LSTM (*Zhang et al.*, 2019b)	64.9K (92.7x)	1.1789s (906.8)
STGAT (*Huang et al.*, 2019)	44.6K (63.7x)	1.3497s (1038.2x)
DAG-Net (*Monti et al.*, 2020)	2.35M (3357.1x)	0.0463s (35.6x)
PIF (*Liang et al.*, 2019)	360.3K (514.7x)	0.1145s (88.1x)
Social GAN (*Gupta et al.*, 2018)	46.3K (66.1x)	0.0968s (74.5x)
Social-STGCNN (*Mohamed et al.*, 2020)	7.6K (11x)	0.0020s (1.5x)
STC-Net	0.8K (1.1x)	0.0015s (1.2x)
STC-Net-NAS	**0.7K**	**0.0013s**

Table 8.8: Model size and efficiency. Only (*Monti et al.*, 2020; *Liang et al.*, 2019; *Gupta et al.*, 2018; *Mohamed et al.*, 2020) achieve a latency that is lower than the frame-rate.

	w/o GC	w/o DTC	w/o FWC	Ours
ADE	0.64	0.59	0.55	**0.51**
FDE	1.02	0.99	0.91	**0.85**
Latency	0.7ms	1.4ms	2.3ms	1.5ms

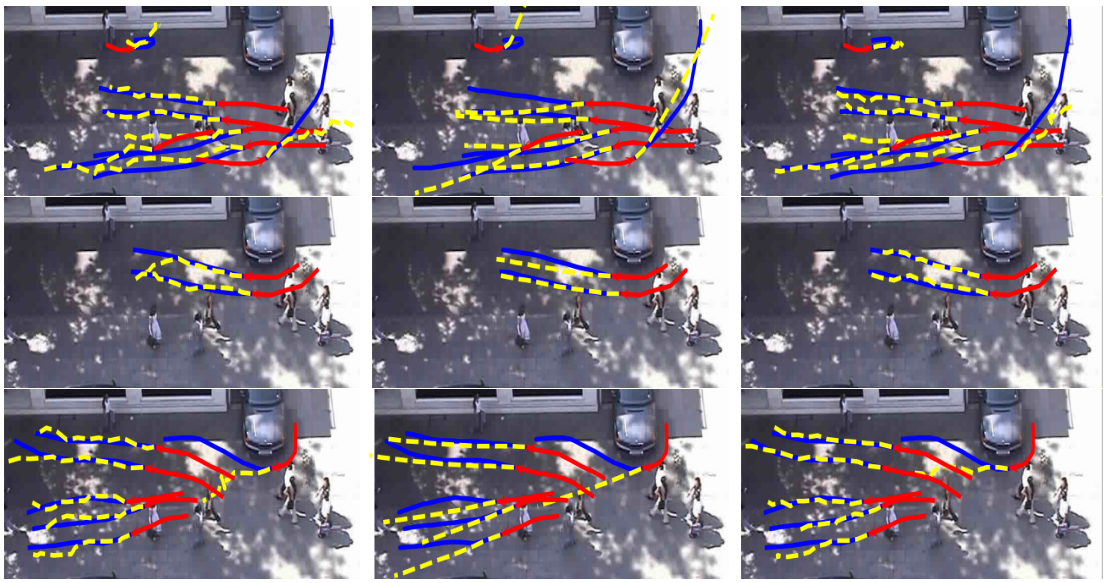
Table 8.9: Impact of the different types of convolutions.

the numbers of *Liang et al.* (2020a) are not comparable since the approach uses additional training data.

Efficiency. Since methods for forecasting trajectories need to be compact and to have a very low inference time, we compare the size and inference time of STC-Net with other state-of-the-art methods in Table 8.8. For measuring the inference time, we used a single 1080Ti GPU. In contrast to previous works, our approach is very compact and comprises less than 1K parameters. From the state-of-the-art methods, only Social-STGCNN (*Mohamed et al.*, 2020) achieves a very low inference time. STC-Net, however, not only achieves a much lower forecasting error than Social-STGCNN, it is even more compact and has a lower latency. It is interesting to note that the neural network search finds an instance that not only makes more accurate predictions, but also reduces the model size and inference time. STC-Net-NAS runs with more than 750 frames per second on a low-budget GPU, yielding a latency of less than 2ms.

Furthermore, we evaluated on the Stanford Drone dataset (image coordinates) the impact of the model size by increasing the number of dilated convolutional layers before the feature-wise convolution and/or the number of channels. The results are shown in Table 8.10. The results show that the error can be further reduced by adding more layers or increasing the number of channels per layer at the cost of increasing the number of parameters or the inference time. Note that for all configurations the approach is still faster than previous works.

Temporal Robustness. We used a sampling rate of 2.5 frames per second, which corresponds to one frame every 0.4 seconds, for training and inference. In Table 8.6, we report the results when we use a lower sampling rate during inference while keeping the number of frames the same. When the sampling rate is higher, we simulate the case when the objects move faster than observed during training. Note that the numbers for different sampling rates are not directly comparable since the sampled trajectories also differ for the different sampling rates, but they indicate how robust the



((a)) Social-STGCNN (*Mohamed et al.*, 2020) ((b)) STGAT (*Huang et al.*, 2019) ((c)) STC-Net

Figure 8.5: Qualitative results. The **red** line is the observed trajectory, the **blue** line is the ground truth of the future trajectory, and the **yellow** dashed line is the prediction.

L	CH	ADE	FED	Params	Inf time
3	1×	11.96	20.12	0.8K	0.0015s
6	1×	11.61	18.79	1.0K	0.0017s
3	2×	11.65	19.79	2.6K	0.0015s
3	4×	11.49	19.04	10.0K	0.0015s
6	4×	**11.34**	**18.65**	13.6K	0.0017s

Table 8.10: Changing the size of the model. From top to bottom, we show the results of the original model, a deeper model, a wider model, and finally a combination of them. L denotes number of layers and CH means increase of number of channels, i.e., K× means increasing the number of channels by K times compared to the original model.

methods are. The results show that our and the other graph-based approaches (*Mohamed et al.*, 2020; *Huang et al.*, 2019) are very robust and can handle objects that are 5 times faster than the ones in the training set. Only for *Monti et al.* (2020), we observe a drastic increase in the error for 2.0 seconds.

Cross-scene Robustness. In Table 8.7, we evaluate how robust the methods are across different scenes. This means that we train the approaches on the Stanford Drone dataset (*Robicquet et al.*, 2016) and evaluate them on the ETH (*Pellegrini et al.*, 2009) and UCY (*Lerner et al.*, 2007) datasets. We use the Stanford Drone dataset for training since it is the largest dataset and contains not only pedestrians but also other objects. Compared with Table 8.1, the average error of Social-STGCNN (*Mohamed et al.*, 2020) and STGAT (*Huang et al.*, 2019) is only slightly higher. While the error increases for most scenes, it even decreases for the HOTEL scene. This is reasonable since the Stanford Drone dataset provides a larger dataset and the trajectories of the HOTEL scenes are relatively simple. This shows that these methods are quite robust to scene changes. Our approach,

however, performs even better and the error even slightly decreases in average. This shows that our approach models the motion very well, is very robust to changes of the sampling rate, and generalizes to new scenes.

Qualitative Analysis. We show some qualitative results in Figure 8.5 and compare our approach to the graph-based approaches Social-STGCNN (*Mohamed et al.*, 2020) and STGAT (*Huang et al.*, 2019). From the images, we can see that the predictions of Social-STGCNN are not smooth and some trajectories even cross which is implausible. The predictions of STGAT are nearly linear, which results in larger errors when the direction changes. In contrast, our method forecasts more plausible trajectories.

8.4 Conclusion

In this chapter, we proposed a highly efficient forecasting model that generates spatially and temporally consistent trajectories without the usage of map information. The network utilizes graph and dilated temporal convolutions to model the spatial and temporal relations of each trajectory. Furthermore, a feature-wise convolution is utilized to forecast trajectories in one pass, and the reconstructed observed and forecast trajectories are jointly refined. By using a neural network architecture search, the network is further optimized. The proposed approach outperforms most other methods in terms of accuracy, is very compact, and achieves a very low latency. In our evaluation, we also demonstrate that the approach is robust to changes in the sampling rate and to scene variations. This makes the proposed approach perfectly suitable for realistic applications.

CHAPTER 9

Aggregation-Interaction Transformer for Efficient Trajectory Forecasting

Although the STC-Net proposed in Chapter. 8 works well for human trajectory prediction which is a low-speed scenario, it is not suitable for autonomous driving where the agents are at high speed. In this case, the map can provide meaningful information as the drivers usually need to follow some driving rules. However, modeling all the interactions between different road users and drivable lanes in an efficient way remains a challenging problem. To address this issue, we propose an efficient architecture that introduces an additional aggregation token per lane and trajectory. It allows to model global dependencies within each lane or trajectory and global dependencies between the lanes or trajectories. We thus term the network Aggregation-Interaction Transformer. As a second contribution, our approach learns intention seeds that are used as queries to generate diverse future trajectories in a very efficient way. We conduct comprehensive experiments on two challenging datasets where our method achieves state-of-the-art performance both in terms of accuracy and efficiency.

Contents

9.1 Introduction

Trajectory forecasting plays an essential role in many applications like mobile robots or autonomous driving. The future trajectory, however, not only depends on the past trajectory of the vehicle itself, but also on the other agents like other road users. The third aspect that has an impact on the future trajectory is the environment.

In the context of autonomous driving, a bird-view map (*Gilles et al.*, 2022; *Phan-Minh et al.*, 2020) can be used to model the nearby environment. A bird-view map, however, is not an efficient representation for providing the information where a vehicle can drive. This is in particular an issue for autonomous driving where the environmental information comes often from a vector-based street map that is also used for navigation. It is therefore very practical to use such compact representation, where vectors represent the drivable lanes, also for trajectory forecasting (*Kim et al.*, 2021; *Gao et al.*, 2020).

However, it remains a major research question how such a representation can be efficiently used for trajectory forecasting. In principle, we need to model the interactions of the agents between them. For instance, not only a vehicle in front of a car has a major impact on the trajectory of the car, but also a vehicle coming from the opposite direction if the car aims to turn left. In the latter case, the other car might be still far away but depending on the speed of the car it can have a strong impact on the future trajectory. Due to the high speed of vehicles, we need to model the global interaction between all agents. In addition, we need to model the interaction between agents and lanes since the lanes define the possible future drivable directions.

The lanes are commonly sub-sampled and convolutional and recurrent neural networks are used to aggregate a global feature for each lane (*Kim et al.*, 2021; *Salzmann et al.*, 2020). The interaction between agents and lanes is then only modeled using these global lane features. This approach, however, is not optimal since RNNs are not capable of modeling long-range dependencies and relations between lanes are not modeled. The latter is very important since lanes are often subdivided into smaller parts and some streets have multiple lanes that can be used. In order to consider spatial relations between lanes, (*Ye et al.*, 2021) consider the sampled points of all lanes as an unstructured 2D point set and use standard point-based methods (*Qi et al.*, 2017d) to process the points. This, however, discards the sequential structure of lanes and an additional module is required to extract sequential information from the points.

In our method, we alleviate the above issues by proposing an efficient transformer architecture for trajectory forecasting. We use the same block both for modeling relations between lanes and between agents. Instead of aggregating the features per lane or trajectory as in *Kim et al.* (2021) or discarding the sequential structure of lanes as in *Qi et al.* (2017d), we aim to model the interactions between all points of all sequences. Since this is computationally infeasible, we introduce an additional aggregation token for each lane or trajectory. The proposed approach then iterates between updating the features of each point of the lane or trajectory and updating only the aggregation token for all lanes or trajectories by modeling the interaction between lanes or trajectories. In this way, the features of the points of the lanes or trajectories are updated based on the entire context of the scene, *i.e.*, taking into account all other lanes or trajectories. Since the steps of aggregation and interaction are repeated, we call it an Aggregation-Interaction Block (AI-Block). The proposed approach is also very practical since the AI-Block provides a unified model that can process spatial (lanes) and temporal sequences (trajectories) in the same way.

As a second contribution, we address the ambiguity of the future and generate multiple predictions for the same observation. A common approach to generating multiple predictions is the use of conditional variational autoencoders, which have been combined with a transformer in (*Yuan et al.*, 2021). In order to generate multiple sequences, (*Yuan et al.*, 2021) requires a diversity sampling technique to generate a latent code and a conditional variational decoder to generate a future trajectory. In this work, we propose an approach that is much more efficient. Instead of a sampling approach,

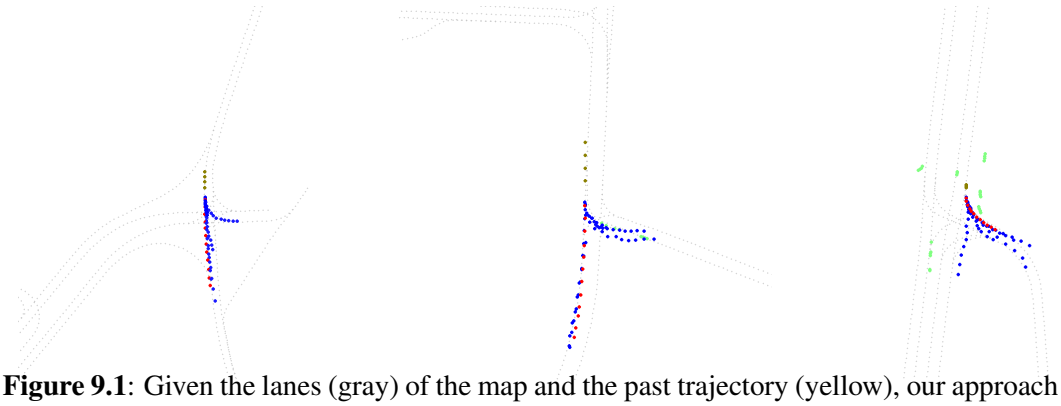


Figure 9.1: Given the lanes (gray) of the map and the past trajectory (yellow), our approach forecasts multiple future trajectories (blue). Red points are ground truth future positions. For better readability, we plot only the five trajectories with the highest score. The approach also considers the trajectories of the other vehicles (green). We can observe that our method can forecast diverse plausible trajectories.

we propose to learn a set of intention seeds, which reflect the different intentions of a driver. These seeds are used as queries and the encoded lanes and trajectories as values and keys. The decoder predicts then for each intention query a future trajectory and a score, which reflects the probability of this trajectory, as it is shown in Figure 9.1. We term the decoder Multi-Intention Decoder since it generates for different learned intention seeds different future trajectories.

We evaluate the proposed approach on the challenging nuScenes (*Caesar et al.*, 2020) dataset where the approach achieves state-of-the-art results. The approach is also very efficient and requires only 1.22 ms for inference, which is 52 times faster than *Yuan et al.* (2021). In summary, the contributions of the work are:

- We propose the Aggregation-Interaction Block, which models global dependencies of points between and within sequences. The block unifies the processing of spatial and temporal sequences.
- We propose the Multi-Intention Decoder, which generates diverse predictions in a very efficient way by learning intention seeds.

9.2 Method

In order to forecast the future trajectory of a vehicle, we need to model the interaction with other road users as well as the available possibilities, which are provided by the drivable lanes. While the information of the road users is provided by a trajectory or temporal sequence for each agent, the lanes are presented as lines or curves in the street map, which are spatial sequences.

Modeling all the interactions of spatial and temporal sequences, however, is non-trivial since there can be many agents and lanes. Furthermore, the position of the vehicle changes over time and the vehicle can change the lane. Figure 9.2 illustrates how lanes are commonly modeled. For instance, RNNs (*Kim et al.*, 2021) have been used to aggregate the information of a single lane. However, RNNs are are not a suitable approach for spatial sequences since they do not capture the global context compared to transformers. Other approaches consider the sampled points of all lanes

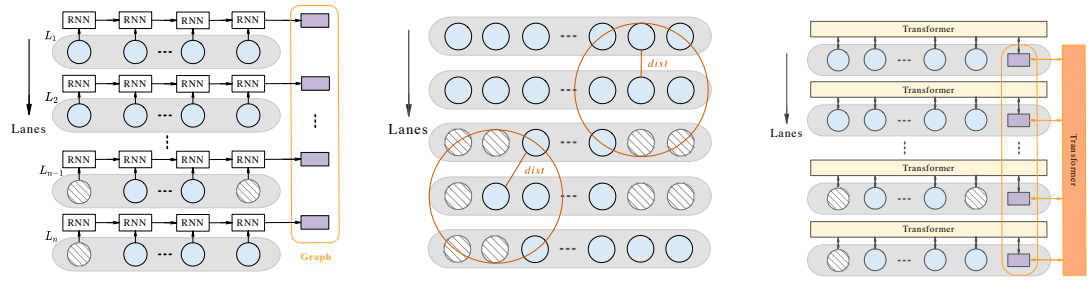


Figure 9.2: Different ways to model lanes. *Left:* RNNs consider lanes like temporal sequences and aggregate the information in sequential order to compute a single feature per lane. RNNs, however, do not model the global context of a lane and they compute the feature without taking other lanes into account. *Middle:* Considering the points of lanes as 2D point cloud allows to model spatial relations between lanes locally, but the specific curve structure of lanes gets lost and there is no global feature per lane. *Right:* Our proposed approach models relations between lanes as well as the relations of points within an entire lane. This is achieved by introducing an additional aggregation token (purple), such that the transformer computes a global feature per lane, and a second transformer that models the global relations between the lanes.

in a map as an unstructured 2D point cloud (*Ye et al.*, 2021). While they model locally the relation of points of different lanes that are spatially close, the lane structure gets lost and the relation is modeled locally at points and not globally at lanes.

We thus propose a new approach that models the relations of lanes globally. While this could be achieved by feeding all sampled points of all lanes into a transformer to compute the attention between all points, this approach is computationally infeasible since there are simply too many points. To address this issue, we introduce an aggregation token per lane to gather the features of each lane and to model the global relations between all lanes as in Figure 9.2. This means that we compute the feature of the aggregation token first for each lane, using all points of a single lane, and then update the aggregation token using the self-attention of all aggregation tokens. These two operations are iterated such that the global context of all lanes is modeled. Besides of being able to model the global context of all lanes and thus the future possibilities a vehicle can move, the approach has the advantage that it can be used for trajectories as well to model the global context along a trajectory and the relations between multiple trajectories. The approach is thus a unified model for spatial (lanes) and temporal (trajectories) sequences.

Before we describe in Section 9.2.2 the block, which we denote by Aggregation-Interaction Block (AI-Block), more in detail, we will introduce some notations in Section 9.2.1. Finally, we describe the full network for trajectory forecasting.

9.2.1 Task Definitaion

Trajectory forecasting for vehicles, as it is defined for datasets like nuScenes(*Caesar et al.*, 2020), requires to forecast the future trajectory of a vehicle. We denote by

$$\mathcal{T}^h = \{x_t : t \in \{-T_h + 1, \dots, 0\}\}$$

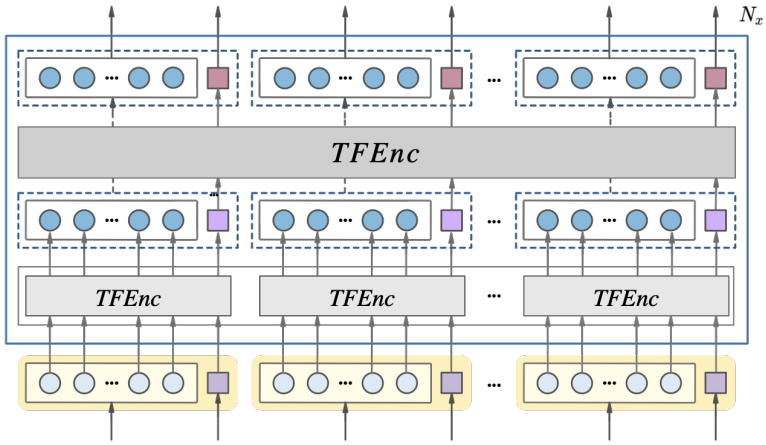


Figure 9.3: Aggregation-Interaction Block (AI-Block). We add to each sequence tokens (circles) an aggregation token (square). The AI-Block computes first the self-attention for each sequence including the aggregation token and updates the features for each token. In the next step, the self-attention is computed across all aggregation tokens (purple square). While the second step only updates the aggregation tokens (red square), the sequence tokens (blue circle) are taken from the previous step. The block is repeated N_x times.

the past ego trajectory of the vehicle where 0 is the current timestamp and x_t is the 2D vehicle position at timestamp t. In addition, the past trajectories of the other road users are provided. We denote them by

$$\mathcal{T}_{1:N} = \{x_t : t \in \{-T_h + 1, \dots, 0\}\}_{1:N}.$$

In our approach, we consider only the closest N vehicles at timestamp 0. If there are less than N other agents, we mask the empty entries in $\mathcal{T}_{1:N}$.

The lanes are denoted by

$$\mathcal{L} = \{x_i : i \in \{1, N_{pt}\}\}_{1:N_l}$$

where each lane is represented by N_{pt} 2D points that are uniformly sampled from its center line. Note that long lanes are split into multiple lanes such that the distance of the sampled points remains the same. For shorter lanes, the missing points are masked as illustrated by the striped circles in Figure 9.2. As for the vehicles, we take only the N_l closest lanes.

While the ground-truth future trajectory of the vehicle is denoted by

$$\mathcal{T}^f = \{x_t : t \in \{1, \dots, T_f\}\}$$

our approach predicts K potential future trajectories to deal with the ambiguity of the future:

$$\hat{\mathcal{T}}^f_{1:K} = \{\hat{x}_t : t \in \{1, \dots, T_f\}\}_{1:K}$$

where a probability score $\hat{s}_k$ is predicted for each future trajectory. The task of trajectory forecasting is thus formulated as:

$$(\hat{\mathcal{T}}^f_k, \hat{s}_k)_{1:K} = \mathbf{Model}(\mathcal{T}^h, \mathcal{T}_{1:N}, \mathcal{L}). \tag{9.1}$$

9.2.2 Aggregation-Interaction Block (AI-Block)

Since the Aggregation-Interaction Block (AI-Block) unifies the processing of spatial and temporal sequences we will apply it to the lanes and trajectories. For the notation, we first discuss the processing of the lanes $\mathcal{L}$ and then the processing of the trajectories $\mathcal{T}^h$ and $\mathcal{T}_{1:N}$.

The AI-Block illustrated in Figure 9.3 takes as input the lanes $\mathcal{L} = \{L_l\}$, where each point is processed by a multilayer perceptron (MLP), and an additional aggregation token A_l that will be used to store the aggregated feature for each lane:

$$F_l = Concat\left(MLP(L_l), A_l\right) \quad \text{for} \quad l = 1, \ldots, N_l. \tag{9.2}$$

The AI-block then updates for each lane the N_{pt} sequence tokens and the aggregation token:

$$F_l^A = TFEnc(PE(F_l)) \quad \text{for} \quad l = 1, \ldots, N_l \tag{9.3}$$

where PE denotes positional encoding (*Vaswani et al.*, 2017b) to maintain the sequential information within each lane and TFEnc is a standard transformer block with self-attention. After this step, not only the features of each point of the lane are updated based on the context of the entire lane, but the aggregation token contains additionally the information of the entire lane.

In the next step, we aim to consider the interaction between the lanes. Since this cannot be computed for all tokens F_l^A, we only take the updated aggregation tokens A_l of every lane and feed them to a second transformer block to compute the self-attention between all lanes and update the aggregation tokens:

$$(A_1, \ldots, A_{N_l}) = TFEnc(A_1, \ldots, A_{N_l}). \tag{9.4}$$

After the interaction step, the new features F_l^I of the tokens are given by

$$F_l^I = Concat\left(\bar{F}_l^A, A_l\right) \quad \text{for} \quad l = 1, \ldots, N_l \tag{9.5}$$

where $\bar{F}_l^A$ denotes the tokens of a lane without aggregation token. The AI-Block is repeated N_x times such that the aggregation and interaction steps are repeated. In our experiments, we use $N_x = 2$.

The AI-Block is applied to the trajectories in the same way. To this end, we concatenate the trajectory of the ego vehicle $\mathcal{T}^h$ with the trajectories of the other road users $\mathcal{T}_{1:N}$

$$\mathcal{T} = Concat(\mathcal{T}^h, \mathcal{T}_{1:N}) \tag{9.6}$$

and feed $\mathcal{T}$ instead of $\mathcal{L}$ to the AI-Blocks. While we denote the lane features after the AI-Block by $F_{\mathcal{L}}^I$, the features of the trajectories are denoted by $F_{\mathcal{T}}^I$.

9.3 Architecture

After having described how the AI-Block works, we now describe the entire network architecture that forecasts the trajectory of a vehicle. As illustrated in Figure 9.4, it consists of an encoder with AI-Blocks for the trajectories and lanes and an decoder that predicts K trajectories $\hat{\mathcal{T}}_k^f$ with scores $\hat{s}_k$. We first describe the full encoder in Section 9.3.1, the decoder in Section 9.3.2, and the loss function in Section 9.3.3.

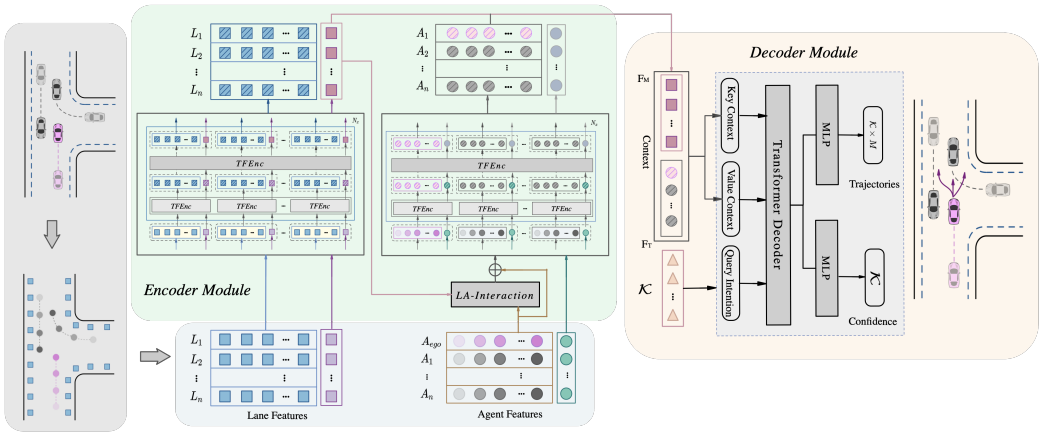


Figure 9.4: Architecture of the proposed method. LA-Interaction denotes the interaction between lanes and trajectory points of the agents.

9.3.1 Encoder

As shown in Figure 9.4, the encoder outputs the features for the lanes $F^I_{\mathcal{L}}$ and the agents $F^I_{\mathcal{T}}$ where each of them models the interactions between lanes and between agents. However, we want to model the interactions between the agents and the lanes not only in the decoder but already in the encoder. We therefore introduce a connection between the output of the encoder for the lanes to the input of the encoder for the agents.

From the output of the encoder of the lanes, we only consider the aggregation tokens $A_{\mathcal{L}}$ whereas $F_{\mathcal{T}}$ denotes the trajectories after the first MLP layer, *i.e.*, $F_{\mathcal{T}} = MLP(\mathcal{T})$. To model the interaction of the trajectory of an agent n with a lane l, we compute the similarity between each trajectory point and each lane:

$$M^n_{t,l} = F^n_t \cdot A_l \quad \text{for} \quad n = 0, \dots, N \tag{9.7}$$

where $\cdot$ denotes the dot product, and F^n_t and A_l are two vectors of the same size corresponding to the feature of the trajectory point t of agent n and the aggregated feature of lane l, respectively. The values of each matrix M are then normalized by

$$M^n_{t,l} = \frac{\exp(M^n_{tl})}{\sum_l \exp(M^n_{tl})}. \tag{9.8}$$

In other words, we compute for each trajectory point normalized weights, which indicate how relevant a lane for this point is. Since the similarity is only based on feature similarity, we also consider spatial proximity by setting $M^n_{t,l} = 0$ if all points x_i of a lane l are too far away from the trajectory point x^n_t. We will evaluate the impact of the spatial proximity in the ablation studies. We then re-normalize the weights such that they sum to 1.

The input of the AI-Block for the trajectories is thus not $F_{\mathcal{T}} = MLP(\mathcal{T})$ with an additional aggregation token A_n for each agent n, but

$$F^n = Concat(F^n + M^n A_{\mathcal{L}}, A_n) \text{ for } n = 0, \dots, N \tag{9.9}$$

where $F^n \in \mathbb{R}^{T_h \times d}$ are the trajectory features from $F_{\mathcal{T}}$ for agent n, $A_{\mathcal{L}} \in \mathbb{R}^{N_l \times d}$ are the aggregation features of all lanes, and $M^n \in \mathbb{R}^{T_h \times N_l}$ is the weight matrix.

	ADE_1	FDE_1	ADE_5	FDE_5	ADE_{10}	FDE_{10}	$MissRate_5$	$MissRate_{10}$	OffRoadRate
CoverNet (*Phan-Minh et al.*, 2020)	3.87	9.26	1.96	-	1.48	-	0.67	-	-
Trajectron++ (*Salzmann et al.*, 2020)	-	9.52	1.88	-	1.51	-	0.70	0.57	0.25
AgentFormer (*Yuan et al.*, 2021)	-	-	1.86	3.89	1.45	2.86	-	-	-
MTP (*Cui et al.*, 2019)	4.42	10.36	2.22	4.83	1.74	3.54	0.74	0.67	0.25
MultiPath (*Chai et al.*, 2019)	4.43	10.16	1.78	3.62	1.55	2.93	0.78	0.76	0.36
ALAN (*Narayanan et al.*, 2021)	4.67	10.0	1.77	3.32	1.10	1.66	0.57	0.45	0.01
MHA-JAM (*Messaoud et al.*, 2021)	3.69	8.57	1.81	3.72	1.24	2.21	0.59	0.45	0.07
GOHOME (*Gilles et al.*, 2022)	-	6.99	1.42	-	1.15	-	0.57	0.47	0.04
P2T (*Deo and Trivedi*, 2020)	-	10.5	1.45	-	1.16	-	0.64	0.46	0.03
LaPred (*Kim et al.*, 2021)	3.51	8.12	1.53	3.37	1.12	2.39	-	-	-
WIMP (*Khandelwal et al.*, 2020)	-	8.49	1.84	-	1.11	-	0.55	0.43	0.04
CXX (*Luo et al.*, 2020a)	-	8.86	1.63	-	1.29	-	0.69	0.60	0.08
SG-Net (*Wang et al.*, 2022a)	-	9.25	1.86	-	1.40	-	0.67	0.52	0.04
AutoBot† (*Girgis et al.*)	-	8.19	1.37	-	1.03	-	0.62	0.44	0.02
PGP‡ (*Deo et al.*, 2022)	-	-	1.30	-	1.00	-	0.61	0.37	0.03
Ours	3.40	7.75	1.32	2.50	0.97	1.55	0.59	0.37	0.02

Table 9.1: Quantitative results on the nuScenes dataset.The following approaches are not directly comparable: † uses an ensemble. ‡ generates more trajectories than other works and clusters them by K-means.

9.3.2 Multi-Intention Decoder

Since there can be multiple plausible future trajectories for the same vehicle, we aim to model this uncertainty, which depends on the unknown intention of the driver. For instance, a driver might drive straight ahead or turn right at a crossing after the traffic light turns green. Without additional information about the intention of the driver, this ambiguity cannot be resolved. We thus predict K future trajectories $\hat{\mathcal{T}}_k^f$ with scores $\hat{s}_k$ that reflect the possibility of each forecast trajectory.

This is achieved by K learnable intention seeds I_k that are used as queries for the decoder. For each query, we get thus a different prediction $\hat{\mathcal{T}}_k^f$. As keys and values, we use the output of the encoder for the lanes and agents. We use only the aggregation tokens $A_{\mathcal{L}}$ from the lanes and only the trajectory tokens without the aggregation tokens, which we denote by $\bar{F}_{\mathcal{T}}^I$, for the trajectories of the agents. While the trajectory tokens are essential for forecasting since they capture the full history of the trajectory, we show in the ablation studies that there is no benefit of using additionally the aggregation token for the trajectories. Vice versa, using all tokens of the lanes increases the inference time but each token of the lanes is not anymore necessary for the decoder as we show in the ablation studies. As illustrated in Figure 9.4, $A_{\mathcal{L}}$ and $\bar{F}_{\mathcal{T}}^I$ are concatenated and we use I_k as query. The decoder block is repeated 2 times and we use two MLPs to finally predict the future trajectory $\hat{\mathcal{T}}_k^f$ and its score $\hat{s}_k$ for each query.

We provide more details of the architecture in the supplemental material and we will release the source code in case of acceptance.

9.3.3 Loss Function

For training the network, we use two loss functions as in *Liang et al.* (2020b). One for the predicted trajectories $\hat{\mathcal{T}}_{1:K}^f$ and one for the predicted scores $\hat{s}_{1:K}$. Since we have only one ground-truth future trajectory $\mathcal{T}^f$ for a single scene, we first detect the predicted trajectory that has the lowest final displacement error (FDE) with respect to the ground-truth trajectory:

$$\hat{k} = \arg\min_k \|x_{T_f} - \hat{x}_{T_f}^k\|_2 \tag{9.10}$$

Method	#Param	Speed	ADE_5	FDE_5	ADE_{10}	FDE_{10}
CoverNet (*Phan-Minh et al.*, 2020)	32.2M	11.69 ms	1.96	-	1.48	-
MTP (*Chai et al.*, 2019)	32.1M	25.59 ms	2.22	4.83	1.74	3.54
Trajectron++ (*Salzmann et al.*, 2020)	0.26M	88.40ms	1.88	-	1.51	-
AgentFormer (*Yuan et al.*, 2021)	0.43M	64.19ms	1.86	3.89	1.45	2.86
Ours (0.25× emb.)	0.13M	0.77ms	1.51	2.99	1.12	1.93
Ours (0.5× emb.)	0.48M	0.84ms	1.47	2.92	1.02	1.69
Ours (0.75× emb.)	1.1M	1.04ms	1.47	2.81	0.99	1.60
Ours (1.0× emb.)	1.9M	1.22ms	1.32	2.50	0.97	1.55
Ours (1.25× emb.)	2.9M	1.60ms	1.38	2.72	0.99	1.59

Table 9.2: Efficiency Analysis. We compare our method to image-based methods (CoverNet (*Phan-Minh et al.*, 2020), MTP (*Chai et al.*, 2019)), an RNN-based method (Trajectron++ (*Salzmann et al.*, 2020)), and a transformer-based method (Agentformer (*Yuan et al.*, 2021)).

where $\hat{x}^k_{T_f}$ is the last point of the predicted trajectory k and x_{T_f} is the last point of the ground-truth trajectory.

Since we aim that the predicted trajectory with the lowest final displacement error has the highest score, we use as in (*Liang et al.*, 2020b) the max-margin loss:

$$L_{score} = \frac{1}{K-1} \sum_{k \neq \hat{k}} \max(0, \hat{s}_k + \varepsilon - \hat{s}_{\hat{k}}), \tag{9.11}$$

i.e., for all other trajectories $k \neq \hat{k}$ the score $\hat{s}_k$ should be by the margin ε lower than the score of the best trajectory $\hat{k}$.

For the forecast trajectories, we use the minimum average displacement error to the ground-truth trajectory with the smooth L1 loss:

$$L_{traj} = \min_k \left\{ \frac{1}{T_f} \sum_{t=1}^{T_f} d(x_t - \hat{x}^k_t) \right\} \tag{9.12}$$

with

$$d(x) = \begin{cases} 0.5x^2 & if \quad ||x|| < 1, \\ ||x|| - 0.5 & otherwise. \end{cases} \tag{9.13}$$

The two loss functions are weighted by $w = 0.4$:

$$L = L_{traj} + w \cdot L_{score}. \tag{9.14}$$

9.4 Experiments

9.4.1 Datasets and Evaluation Metrics

We evaluate our method on widely used datasets for trajectory forecasting: nuScenes (*Caesar et al.*, 2020). It provides high-definition (HD) maps associated with the trajectory data. The nuScenes dataset includes 245,414 trajectory instances across 1,000 different scenes. Each trajectory instance lasts for 8 seconds sampled at 2Hz. The trajectory forecasting task defined in the nuScenes dataset

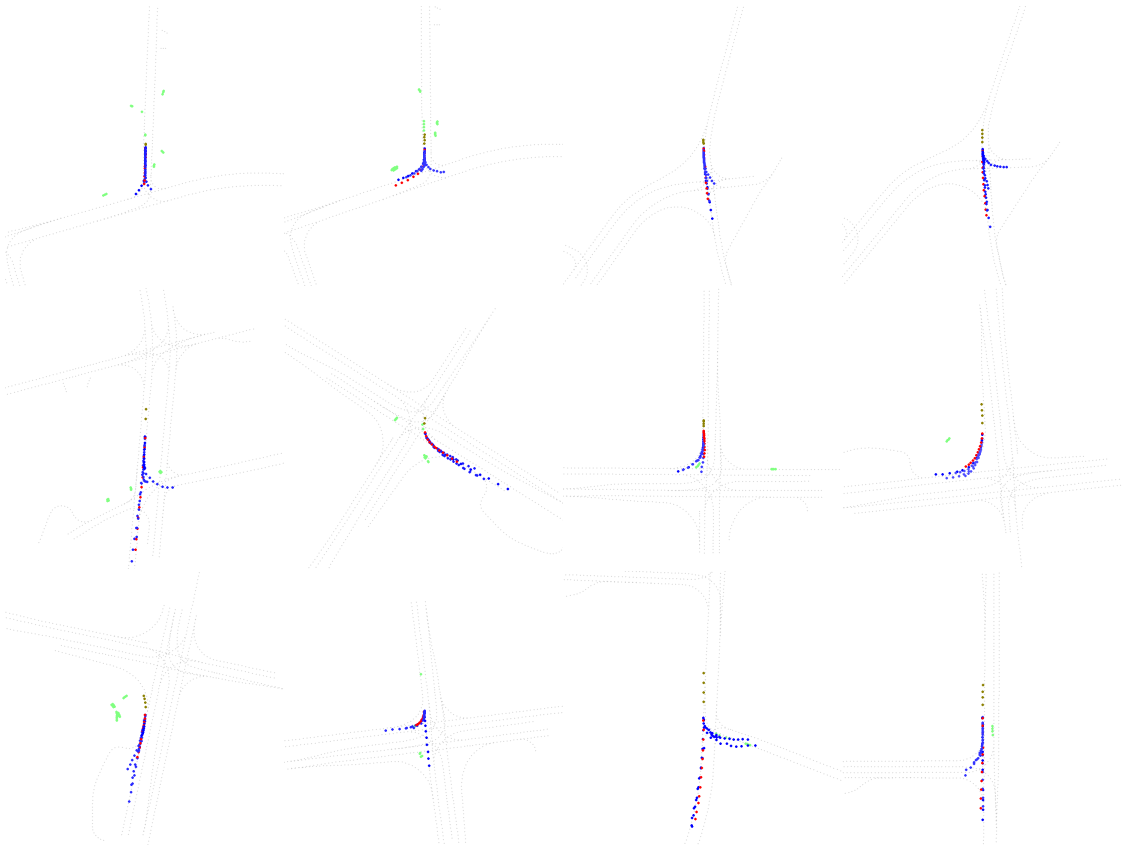


Figure 9.5: Qualitative results for the Nuscenes dataset. Best seen using the zoom function of the PDF reader. The small gray dots denote the sampled lanes. Yellow dots denote the past trajectory. Red dots denote the future ground-truth trajectory. Blue dots denote the predicted trajectories where a darker color corresponds to a higher score. Green dots are trajectories of other agents. Our method generates diverse yet plausible predictions.

requires to forecast the future trajectory of the ego vehicle for 6 seconds given the first 2 seconds as observed trajectory. As in previous works (*Kim et al.*, 2021), we train our approach on the training set and evaluate its performance on the validation set.

We report the widely used minimum average displacement error ADE_K and the final displacement error FDE_K of K forecast trajectories with respect to the ground-truth trajectory $\mathcal{T}^f$:

$$ADE_K(\hat{\mathcal{T}}^f_{1:K}, \mathcal{T}^f) = \min_k \left\{ \frac{1}{T_f} \sum_{t=1}^{T_f} ||x_t - \hat{x}^k_t||_2 \right\}, \tag{9.15}$$

$$FDE_K(\hat{\mathcal{T}}^f_{1:K}, \mathcal{T}^f) = \min_k ||x_{T_f} - \hat{x}^k_{T_f}||_2. \tag{9.16}$$

We also report the Miss Rate for different values of K and the Off Road Rate for the nuScenes dataset (*Caesar et al.*, 2020). The former metric is defined as the proportion of misses over all agents. If the maximum L2 distance between the best prediction and the ground truth trajectory is larger than 2m, this prediction is defined as a miss. Similarly, if a prediction does not entirely lie in the drivable area, it is defined as off road. The Off Road Rate is defined as the proportion of off road predictions over all agents.

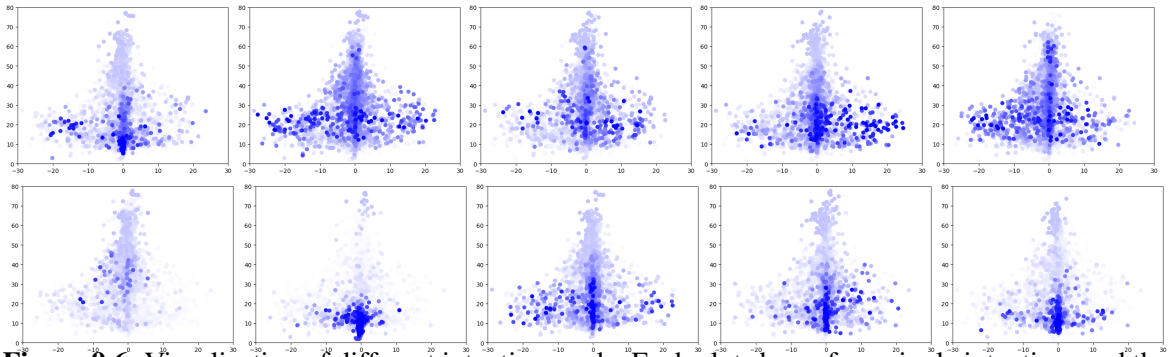


Figure 9.6: Visualization of different intention seeds. Each plot shows for a single intention seed the endpoints of all predicted trajectories. A darker color means a higher score.

9.4.2 Comparison to State-of-the-Art

We first compare our method to previous state-of-the-art methods on the nuscens dataset. The results are shown in Tables 9.1. From Table 9.1, we can observe that our method achieves the best performance for almost all metrics apart from FDE_1 and MissRate_5. For the FDE_1 metric, GOHOME (*Gilles et al.*, 2022) achieves the best result. GOHOME predicts a heatmap representing the spatial distribution of the final trajectory point and then uses a sampling algorithm to generate the trajectory, which optimizes the Miss Rate for the trajectory. While this approach is optimized for low FDE_1 and MissRate_5, it is outperformed by other methods for other metrics. The MissRate_5 of our method is very close to the best performance (*Khandelwal et al.*, 2020) and for MissRate_{10} our method outperforms all other approaches by a large margin. In particular, the results for ADE_K show that our approach forecasts very accurate trajectories.

Apart from quantitative results, we also show some qualitative results in Figure 9.5. From these examples, we can observe that our method can produce diverse plausible predictions. Especially at crossroads, the predictions of our method can cover almost all possible future situations, like driving straight and turning left or right.

The efficiency of approaches for trajectory forecasting is an important factor for autonomous driving. We therefore compare the efficiency of our method with other methods in Table 9.2. All experiments have been conducted on a single Nvidia Titan RTX GPU. We can see that the methods that directly process images of the HD map are not very efficient like CoverNet (*Phan-Minh et al.*, 2020) or MTP (*Cui et al.*, 2019). While Trajectron++ (*Salzmann et al.*, 2020) and Agentformer (*Yuan et al.*, 2021) use a compact vector representation of the lanes and thus require less parameters, their inference time is even higher. The RNN-based approach (*Salzmann et al.*, 2020) suffers from the low efficiency of RNNs, which process data sequentially. The approach (*Yuan et al.*, 2021) uses a combination of a conditional variational autoencoder and a transformer to generate multiple trajectories, which is by far less efficient than our Multi-Intention Decoder which uses learned intention seeds to generate multiple trajectories. As a comparison, our approach is 52 times faster than Agentformer and 72 times faster than Trajectron++. If the size of the model is very important, we can reduce the feature dimensionality of the embeddings. If we reduce the dimensionality of the embedding by 0.25, the size of the network is smaller than *Salzmann et al.* (2020) but the forecast trajectories are still more accurate.

Agg	R	ADE_5	FDE_5	ADE_{10}	FDE_{10}
w/o map	-	1.84	3.96	1.26	2.41
w/o LA-Inter.	-	1.39	2.69	0.99	1.60
M_{binary}	2m	1.57	3.10	1.00	1.60
	3m	1.43	2.78	1.00	1.60
	4m	1.45	2.80	0.99	1.58
	5m	1.47	2.81	1.01	1.60
	$+\infty$	3.75	6.27	1.94	2.71
M_{dist}	2m	1.51	2.96	1.01	1.64
	3m	1.52	2.96	1.00	1.62
	4m	1.45	2.81	1.01	1.63
	5m	1.48	2.87	1.02	1.65
	$+\infty$	1.53	3.71	1.04	1.64
M	2m	1.38	2.64	0.99	1.58
	3m	1.40	2.70	0.98	1.57
	4m	**1.32**	**2.50**	**0.97**	**1.55**
	5m	1.36	2.62	0.98	1.58
	$+\infty$	1.45	2.82	0.99	1.61

Table 9.3: Impact of the attention matrix M for modeling the interaction between agents and lanes in the encoder.

w	ADE_1	FDE_1	ADE_5	FDE_5	ADE_{10}	FDE_{10}
0.1	3.61	8.28	1.33	2.79	0.94	1.57
0.2	3.97	9.13	1.39	2.68	0.99	1.58
0.3	3.44	7.90	1.48	2.88	0.98	1.56
0.4	3.40	7.75	1.32	2.50	0.97	1.55
0.5	3.42	7.85	1.50	2.94	0.98	1.58
0.6	3.44	7.82	1.48	2.91	0.99	1.59

Table 9.4: Impact of w in Equation (14). Results are reported for the nuScenes dataset.

9.4.3 Ablation Studies

We perform the ablation studies on the nuScenes dataset. As described in Section. 9.3.1, we compute the attention matrix M in Equation 9.7 between agents and lanes, which is then used to update the features of the trajectories of the agents in Equaton. 9.9. The interaction is only considered for a point on a trajectory when the closest point of a lane is within a radius R. In Table 9.3, we report the results for different values of R. The results show that the impact of R is relatively small, but the best performance is achieved for $R = 4m$, which we use in all experiments. However, if we do not consider spatial proximity and use $R = \infty$, the error clearly increases.

We also evaluate two variants where we use only the spatial proximity but we do not compute the similarity between each trajectory point and each lane in Equation 9.7. In case of M_{binary}, all lanes that are within the radius R of a trajectory point have the same weight. In case of M_{dist}, we compute the values of the matrix M based on the inverse distance of a lane to a trajectory point. We can observe that M_{binary} and M_{dist} perform worse than M. This shows that using only spatial proximity is insufficient. We also evaluate a setup where the interactions between agents and lanes are only modeled in the decoder but not in the encoder, *i.e.*, we remove the connection between the output of the lane encoder to the input of the agent encoder shown in Figure 9.4. We denote this setting by 'w/o LA-Inter.'. The errors are higher than in the proposed setting but the results are still good since the interactions between lanes and agents are still considered in the decoder. When we do

Agent	Lane	ADE_1	FDE_1	ADE_5	FDE_5	ADE_{10}	FDE_{10}	Speed
TCN	TF	5.11	11.15	1.69	3.20	1.13	1.79	1.19 ms
LSTM		4.33	9.88	1.47	2.83	1.02	1.60	1.22 ms
Bi-LSTM		5.82	13.14	1.51	2.95	1.03	1.68	1.25 ms
TF	TF	3.40	7.75	1.32	2.50	0.97	1.55	1.22 ms
TF	TCN+ Maxpool	3.73	8.62	1.59	3.21	1.14	2.03	0.44 ms
	TCN + Meanpool	3.63	8.40	1.53	3.07	1.11	1.93	0.44 ms
	LSTM	4.49	10.43	1.46	2.89	1.01	1.68	1.51 ms
	Bi-LSTM	3.64	8.39	1.48	2.93	1.01	1.67	2.85 ms

Table 9.5: Different variants of the agent and lane encoder.

not consider the interactions between agents and lanes, *i.e.*, we remove the lane encoder completely ('w/o map'), the errors increase drastically.

In Table 9.6, we evaluate the impact of the tokens that we feed from the lane and agent encoder to the decoder. The first row shows the result of the proposed network, *i.e.*, we use only the aggregation tokens from the lane encoder and the tokens of the trajectory points from the agent encoder. If we use the aggregation token from the agent encoder, in addition, the results are similar. However, the results are worse when only the aggregation token from the agent encoder is used. This is expected since the points encode the full history of the trajectory, which is needed for forecasting. For the lane encoder, it is the other way around. Using all points of the lanes increases substantially the inference time due to the large number of lane points. However, they do not decrease the error. This is reasonable as not all lane points contribute to the final prediction and including such redundant information can mislead the final prediction. Finally, we report the errors when only the features from the agent encoder are used. Note that the errors are much lower compared to 'w/o map' in Table 9.3 since the information of the lanes is still used as input to the agent encoder. While using the tokens of the trajectory points performs in the last setting best as before, it also shows that features both from the lane and agent encoder are needed for the decoder.

Both for the agent and lane encoder, we use AI-Blocks, which are based on transformers (TF). In Table 9.5, we evaluate other implementations. For the agent encoder, we replace the transformer encoder by a temporal convolution network (TCN), RNN with long short-term memory (LSTM), and a bidirectional RNN with long short-term memory (Bi-LSTM). All variants have a similar inference time as TF, but the error is much higher, especially ADE_1 and FDE_1.

Since we use only the aggregated feature of each lane as input to the decoder, we use max-pooling or mean-pooling at the end of the TCN to aggregate the features of all points of a lane. LSTM performs worst for the lane encoder. Compared to LSTM, Bi-LSTM achieves a lower error but at the cost of higher inference time. Note that the differences in inference time for the lane encoder are larger than for the agent encoder since there are much more lane points than trajectory points of agents. We also observe that LSTM performs better than Bi-LSTM for the agent encoder, but much worse than Bi-LSTM for the lane encoder. This shows that Bi-LSTM is useful for processing spatial (lane) sequences, but not temporal (agent) sequences. TCN with mean-pooling is very fast but the error is higher compared to TF and ADE_{10} and FDE_{10} are even higher than for LSTM and Bi-LSTM.

We also evaluate the impact of parameter w in Equation (14) in Table 9.4. In all other experiments, we use $w = 0.4$.

Finally, we visualize the endpoint of each generated trajectory for each intention seed in Figure 9.6 where a darker color means a higher score. We only show the endpoints of the forecast

Lane	Agent	Speed	ADE_5	FDE_5	ADE_{10}	FDE_{10}
Token	Point	1.22ms	**1.32**	**2.50**	**0.97**	**1.55**
Token	Token	1.20ms	1.78	3.28	1.15	1.79
Token	Token + Point	1.22ms	1.35	2.57	0.98	1.55
Point	Point	3.87ms	1.54	2.99	1.01	1.64
Token + Point	Point	3.96ms	1.42	2.73	1.00	1.61
	Point	1.15ms	1.46	2.87	1.00	1.64
	Token	1.14ms	1.50	2.98	1.05	1.75
	Point + Token	1.15ms	1.49	2.97	1.01	1.65

Table 9.6: Impact of the context information for the decoder. While we use from the lane encoder only the aggregation tokens and from the agent encoder the tokens of the trajectories without aggregation token (first row), we evaluate different other combinations including a setting where only the tokens of the agent encoder are used (last three rows).

trajectories. We observe that different learned seeds generate different trajectories.

9.5 Conclusion

In this chapter, we proposed an efficient architecture for trajectory forecasting. It models the interactions of points within sequences and between sequences by introducing an additional aggregation token per sequence. The so-called Aggregation-Interaction Block iterates between computing the attentions within sequences including the aggregation token and globally between sequences using only the aggregation tokens. The block unifies the processing of spatial and temporal sequences. In order to generate multiple future trajectories for the same observation, we propose to learn intention seeds that are used as queries. We have evaluated the proposed approach on two challenging datasets and shown that it forecasts diverse plausible trajectories. The approach achieves on both datasets state-of-the-art performance and requires only 1.22 ms for inference. Compared to STC-Net presented in Chapter 8, the improvement of this method mainly comes from two factors. First, it is with the ability to utilize the map information. This becomes more important under autonomous driving scenarios as the drivers need to follow some traffic guidelines and lane provide strong prior information. In the autonomous driving scenario, the driving behavior varies dramatically in a short time, like acceleration. With a better temporal modeling capacity, the proposed method can outperform STC-Net.

CHAPTER 10

Conclusion

In this chapter, we summarize the contributions of the thesis for efficient perception and forecasting for autonomous vehicles. Furthermore, we discuss future directions for research to advance state-of-the-art approaches for scene understanding and fine-grained motion forecasting.

Contents

10.1 Summary

In this thesis, we proposed approaches that achieve efficient LiDAR point cloud semantic segmentation and motion forecasting. Compared to previous state-of-the-art methods, our proposed methods achieve a good balance between performance and efficiency which makes them quite suitable for realistic applications.

First, we proposed a highly efficient projection-based architecture for LiDAR point cloud semantic segmentation. Apart from semantic information, moving information is also crucial for environment perception. Thus we proposed a general projection-based LiDAR data representation that can compress moving information. With this data representation and the above projection-based architecture, our method can achieve efficient moving object segmentation. Furthermore, we investigated the relationship among mainstream LiDAR point cloud semantic segmentation methods, including point-based, projection-based, and voxel-based methods. For point-based methods, we proposed a general projection strategy on operations in point-based methods, including sampling, grouping, and propagation. With this projection strategy, we can convert any point-based methods into projection-based methods thus improving both efficiency and performance a lot. To improve projection-based methods to achieve the same performance as voxel-based methods, we introduce temporal consistency in both training and inference. Apart from scene understanding, motion forecasting is also highly important for the safety of autonomous driving. To this end, we propose an efficient architecture that can produce accurate while diverse outputs no matter if the map information is available or not.

In the following, we will discuss these contributions in detail.

10.1.1 LiDAR point cloud semantic segmentation

Prior methods aimed at semantic segmentation of LiDAR point clouds typically prioritize accuracy, often overlooking efficiency which is pivotal for real-world applications. To address this, in Chapter 4, we introduce a highly efficient architecture designed to process 2D projection maps derived from input LiDAR sensors. By leveraging the inherent characteristics of LiDAR sensors and transitioning sparse 3D data into a dense 2D format, our approach strikes a favorable balance between accuracy and efficiency. In Chapter 5, we build upon the aforementioned architecture to ascertain the motion status of each point within the LiDAR point cloud. This is accomplished by employing a generalized projection-based LiDAR data representation capable of encapsulating motion information. Moreover, we have established a new benchmark for LiDAR-driven moving object segmentation, grounded on the SemanticKITTI dataset.

The discussions above predominantly center on projection-based techniques. In Chapter 6, we delve into enhancing point-based methods for semantic segmentation of LiDAR point clouds. We unveil a broad concept of re-engineering 3D point-based operations to function within the projection space. This innovative design facilitates the transformation of any point-based methods into projection-based ones, significantly boosting their accuracy and efficiency for LiDAR point cloud semantic segmentation. A notable challenge with projection-based methods is the discernible disparity between them and voxel-based techniques. In Chapter 7, we identify that this disparity stems from the "many-to-one" issue, attributed to the limited horizontal and vertical angular resolution of the range image. To address this, we introduce a temporal fusion layer to glean valuable insights from previous scans, amalgamating them with the current scan. Additionally, we propose a max-voting-based post-processing strategy to rectify incorrect predictions.

10.1.2 Motion Forecasting

Apart from scene understanding, accurate and efficient trajectory forecasting is also of vital importance for the safety of autonomous driving. As the safety of humans is with the highest priority, we first work on mapless human trajectory prediction which is demonstrated in Chapter 8. We propose an efficient architecture to enforce the temporal consistency between historical data and predicted future data. Although this method performs well for low-speed scenarios, it is not suitable for autonomous driving where the agents are at high speed. In this case, the map can provide meaningful information as the drivers need to follow some driving rules. One issue here is that modeling all the interactions between different road users and drivable lanes efficiently remains a challenging problem. To address this issue, in Chapter 9, we propose an efficient architecture that introduces an additional aggregation token per lane and trajectory. It allows to modeling global dependencies within each lane or trajectory and global dependencies between the lanes or trajectories. We thus term the network Aggregation-Interaction Transformer. As a second contribution, our approach learns intention seeds that are used as queries to generate diverse future trajectories in a very efficient way.

10.2 Future Work

In this thesis, I have treated 3D scene understanding and motion forecasting as two separate tasks. In future research, these two tasks can be merged into a unified framework to achieve precise scene-level forecasting. To achieve this goal, the implicit representation techniques can be leveraged for 3D

scene representation. Thus the whole target can be splited into two subtasks: decomposed implicit representation for large-scale dynamic scenes and fine-grained agent action forecasting.

10.2.1 Decomposed implicit representation for large-scale dynamic scenes

Forecasting tasks play a crucial role in various domains, such as Autonomous Driving and Human-Object Interaction (HOI). However, previous approaches have primarily emphasized object-level tasks, such as trajectory prediction and object interaction anticipation. In recent years, there has been a notable shift in research focus towards scene-level forecasting, encompassing occupancy forecasting and future frame prediction (*Pourheydari et al.*). These scene-level forecasting tasks provide intricate future information, making them inherently more challenging.

There has been a notable shift in 3D vision from explicit representations, such as point clouds, meshes, and voxels, to implicit representations like NeRF (*Mildenhall et al.*, 2021) in recent years. Implicit representations offer greater flexibility and efficient reconstruction of continuous scenes compared to explicit representations. However, since implicit representations encode 3D scenes using MLPs without explicit 3D structures, most existing scene-level forecasting methods cannot be directly applied to them. One potential strategy is to store the entire scene as a 4D representation (*Pumarola et al.*, 2021) and then make predictions based on observations. However, this strategy may yield challenging results due to the lack of constraints, such as ensuring that an observed car remains recognizable as a car in the prediction.

To address the aforementioned challenges, the entire scene can be divided into two categories: objects (things) and backgrounds (stuff), with only objects being movable. To efficiently handle this, a potential solution is to decompose the scene and represent objects and backgrounds separately. For handling backgrounds, a key challenge arises due to the often substantial scale of the scene, which may require significant computation resources. To tackle this issue, efficient structures can be employed, such as Instant-NGP (*Müller et al.*, 2022), to store the entire scene in a computationally optimized manner. Regarding objects, each object can be modeled by using a dynamic implicit representation. This approach can capture the intricate and diverse movements of individual objects effectively. By employing these strategies, the above challenges related to scene scale can be overcome.

To make scene decomposition possible, objects need to be separated from backgrounds. This can be achieved by incorporating semantic information into implicit representations. Several existing methods (*Zhi et al.*, 2021) incorporate a pre-trained 2D architecture to extract a 2D feature map containing semantic information, as demonstrated in *Kirillov et al.* (2023). These methods then utilize implicit representations to predict these features. However, the use of pre-trained 2D architectures lacks awareness of multi-view constraints, which can lead to suboptimal results. To address this concern, the future solutions need to align these extracted features before performing fusion.

10.2.2 Fine-grained agent action forecasting

As mentioned earlier, the majority of forecasting methods primarily concentrate on instance-level forecasting, which involves predicting the trajectory of each individual object. While this approach suffices for rigid objects like cars, it falls short when dealing with non-rigid objects, such as pedestrians. The reason is that by focusing solely on instance-level motion forecasting, a considerable amount of valuable information is lost. One crucial aspect of non-rigid objects, like pedestrians, is

that their motion can reveal important future context and actions, such as waiting or turning. However, this valuable contextual information is overlooked in instance-level prediction.

In the previous chapters, the objects are extracted from the entire scene and represent them separately. This simplifies the task of forecasting fine-grained motions in the future, the future motion of each object can be predicted individually. This is in contrast to attempting to forecast the entire scene or all objects' motions simultaneously, which can be a more challenging task. Under this setting, information about the nearby environment is encoded for each agent (object) in the scene. The predictions are conducted based on both scene information and agent history information.

Bibliography

Akan, Adil Kaan and Güney, Fatma. Stretchbev: Stretching future instance prediction spatially and temporally. In *Proc. of the Europ. Conf. on Computer Vision*, pages 444–460. Springer, 2022. (Cited on page 15.)

Al-Rfou, Rami; Choe, Dokook; Constant, Noah; Guo, Mandy, and Jones, Llion. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3159–3166, 2019. (Cited on page 25.)

Alahi, Alexandre; Goel, Kratarth; Ramanathan, Vignesh; Robicquet, Alexandre; Fei-Fei, Li, and Savarese, Silvio. Social lstm: Human trajectory prediction in crowded spaces. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 961–971, 2016. (Cited on pages 13, 14, 95, 99, 100 and 103.)

Alnaggar, Yara Ali; Afifi, Mohamed; Amer, Karim, and ElHelw, Mohamed. Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1800–1809, 2021. (Cited on page 12.)

Alonso, Iñigo; Riazuelo, Luis; Montesano, Luis, and Murillo, Ana C. 3D-MiniNet: Learning a 2D representation from point clouds for fast and efficient 3D LIDAR semantic segmentation. *arXiv preprint arXiv:2002.10893*, 2020. (Cited on page 12.)

Amirian, Javad; Hayet, Jean-Bernard, and Pettré, Julien. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *CVPR Workshops*, pages 0–0, 2019. (Cited on pages 99 and 100.)

Ando, Angelika; Gidaris, Spyros; Bursuc, Andrei; Puy, Gilles; Boulch, Alexandre, and Marlet, Renaud. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. In *CVPR*, 2023. (Cited on pages 13, 85 and 86.)

Bahdanau, Dzmitry; Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 22.)

Behley, J. and Stachniss, C. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems*, 2018. URL `http://www.roboticsproceedings.org/rss14/p16.pdf`. (Cited on pages 55 and 59.)

Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C., and Gall, J. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2019a. URL `https://arxiv.org/pdf/1904.01416.pdf`. (Cited on pages 51 and 55.)

Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Gall, J., and Stachniss, C. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI

Dataset. *The International Journal on Robotics Research*, 40(8-9):959–967, 2021a. (Cited on page 82.)

Behley, Jens; Steinhage, Volker, and Cremers, Armin B. Efficient radius neighbor search in three-dimensional point clouds. In *IEEE International Conference on Robotics and Automation*, pages 3625–3630, 2015. (Cited on page 68.)

Behley, Jens; Garbade, Martin; Milioto, Andres; Quenzel, Jan; Behnke, Sven; Stachniss, Cyrill, and Gall, Juergen. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2019b. (Cited on page 15.)

Behley, Jens; Garbade, Martin; Milioto, Andres; Quenzel, Jan; Behnke, Sven; Stachniss, Cyrill, and Gall, Juergen. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019c. (Cited on pages 82, 83 and 86.)

Behley, Jens; Garbade, Martin; Milioto, Andres; Quenzel, Jan; Behnke, Sven; Stachniss, Cyrill, and Gall, Jurgen. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 9297–9307, 2019d. (Cited on pages vii, 29, 41, 65 and 74.)

Behley, Jens; Garbade, Martin; Milioto, Andres; Quenzel, Jan; Behnke, Sven; Gall, JÃ¼rgen, and Stachniss, Cyrill. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The SemanticKITTI dataset. *The International Journal of Robotics Research*, 2021b. (Cited on page 41.)

Berman, Maxim; Rannen Triki, Amal, and Blaschko, Matthew B. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018a. (Cited on pages 40 and 73.)

Berman, Maxim; Triki, Amal Rannen, and Blaschko, Matthew B. The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018b. (Cited on page 86.)

Bhattacharyya, Apratim; Hanselmann, Michael; Fritz, Mario; Schiele, Bernt, and Straehle, Christoph-Nikolas. Conditional flow variational autoencoders for structured sequence prediction. *arXiv preprint arXiv:1908.09008*, 2019. (Cited on page 101.)

Bi, Huikun; Zhang, Ruisi; Mao, Tianlu; Deng, Zhigang, and Wang, Zhaoqi. How can i see my future? fvtraj: Using first-person view for pedestrian trajectory prediction. In *ECCV*, 2020. (Cited on pages 14 and 99.)

Bokhovkin, Alexey and Burnaev, Evgeny. Boundary loss for remote sensing imagery semantic segmentation. In *ISNN*, 2019. (Cited on page 86.)

Caesar, Holger; Bankiti, Varun; Lang, Alex H; Vora, Sourabh; Liong, Venice Erin; Xu, Qiang; Krishnan, Anush; Pan, Yu; Baldan, Giancarlo, and Beijbom, Oscar. nuScenes: A multimodal dataset for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. (Cited on pages x, 30, 32, 74, 77, 109, 110, 115 and 116.)

Camuffo, Elena; Mari, Daniele, and Milani, Simone. Recent advancements in learning algorithms for point clouds: An updated overview. *Sensors*, 22(4):1357, 2022. (Cited on pages vii and 29.)

Cao, Anh-Quan and de Charette, Raoul. Monoscene: Monocular 3d semantic scene completion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 3991–4001, 2022. (Cited on page 15.)

Chai, Yuning; Sapp, Benjamin; Bansal, Mayank, and Anguelov, Dragomir. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. (Cited on pages xiv, 114 and 115.)

Chaofan Tao, QJ and Luo, P. Dynamic and static context-aware lstm for multi-agent motion prediction. In *ECCV*, 2020. (Cited on page 99.)

Chen, Liang-Chieh; Papandreou, George; Schroff, Florian, and Adam, Hartwig. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. (Cited on page 34.)

Chen, Liang-Chieh; Zhu, Yukun; Papandreou, George; Schroff, Florian, and Adam, Hartwig. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. of the Europ. Conf. on Computer Vision*, pages 801–818, 2018. (Cited on pages 43, 47, 75, 76 and 78.)

Chen, X.; Milioto, A.; Palazzolo, E.; GiguÃ¨re, P.; Behley, J., and Stachniss, C. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2019a. URL `http://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/chen2019iros.pdf`. (Cited on page 59.)

Chen, X.; Läbe, T.; Milioto, A.; Röhling, T.; Vysotska, O.; Haag, A.; Behley, J., and Stachniss, C. OverlapNet: Loop Closing for LiDAR-based SLAM. In *RSS*, 2020. (Cited on page 13.)

Chen, X.; Läbe, T.; Milioto, A.; Röhling, T.; Behley, J., and Stachniss, C. OverlapNet: A Siamese Network for Computing LiDAR Scan Similarity with Applications to Loop Closing and Localization. In *Autonomous Robots*, 2021a. (Cited on page 13.)

Chen, Xieyuanli; Palazzolo, Andres Milioto Emanuele; Giguère, Philippe; Behley, Jens, and Stachniss, C. Suma++: Efficient lidar-based semantic slam. In *IROS*, 2019b. (Cited on page 85.)

Chen, Xieyuanli; Li, Shijie; Mersch, Benedikt; Wiesmann, Louis; Gall, Jürgen; Behley, Jens, and Stachniss, Cyrill. Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data. *IEEE Robotics and Automation Letters*, 6(4):6529–6536, 2021b. (Cited on pages 7 and 48.)

Chen, Xieyuanli; Li, Shijie; Mersch, Benedikt; Wiesmann, Louis; Gall, Jürgen; Behley, Jens, and Stachniss, Cyrill. Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data. In *R-AL*, 2021c. (Cited on pages 13 and 84.)

Chen, Yuxiao; Ivanovic, Boris, and Pavone, Marco. Scept: Scene-consistent, policy-based trajectory predictions for planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17103–17112, 2022. (Cited on page 15.)

Cheng, Hao; Liao, Wentong; Tang, Xuejiao; Yang, Michael Ying; Sester, Monika, and Rosenhahn, Bodo. Exploring dynamic context for multi-path trajectory prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12795–12801. IEEE, 2021a. (Cited on page 14.)

Cheng, Hui-Xian; Han, Xian-Feng, and Xiao, Guo-Qiang. Cenet: Toward concise and efficient lidar semantic segmentation for autonomous driving. In *ICME*, 2022. (Cited on pages 85, 86, 87, 88 and 89.)

Cheng, Ran; Razani, Ryan; Taghavi, Ehsan; Li, Enxu, and Liu, Bingbing. 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12547–12556, 2021b. (Cited on page 11.)

Chitta, Kashyap; Prakash, Aditya, and Geiger, Andreas. Neat: Neural attention fields for end-to-end autonomous driving. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 15793–15803, 2021. (Cited on page 15.)

Choy, Christopher; Gwak, JunYoung, and Savarese, Silvio. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. (Cited on pages 88 and 89.)

Cortinhal, Tiago; Tzelepis, George, and Aksoy, Eren Erdal. SalsaNext: Fast, Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2020a. URL `https://arxiv.org/pdf/2003.03653.pdf`. (Cited on pages 51, 54, 55 and 57.)

Cortinhal, Tiago; Tzelepis, George, and Aksoy, Eren Erdal. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *ISVC*, 2020b. (Cited on page 87.)

Cortinhal, Tiago; Tzelepis, George, and Erdal Aksoy, Eren. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15*, pages 207–222. Springer, 2020c. (Cited on page 11.)

Cui, Henggang; Radosavljevic, Vladan; Chou, Fang-Chieh; Lin, Tsung-Han; Nguyen, Thi; Huang, Tzu-Kuo; Schneider, Jeff, and Djuric, Nemanja. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. (Cited on pages 14, 114 and 117.)

Dendorfer, Patrick; Os̆ep, Aljos̆a, and Leal-Taix'e, Laura. Goal-gan: Multimodal trajectory prediction based on goal position estimation. In *ACCV*, 2020. (Cited on pages 14 and 99.)

Deo, Nachiket and Trivedi, Mohan M. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020. (Cited on pages 101 and 114.)

Deo, Nachiket; Wolff, Eric, and Beijbom, Oscar. Multimodal trajectory prediction conditioned on lane-graph traversals. In *Conference on Robot Learning*, pages 203–212. PMLR, 2022. (Cited on page 114.)

Ding, Ben. Lenet: Lightweight and efficient lidar semantic segmentation using multi-scale convolution attention. In *arXiv*, 2023. (Cited on page 85.)

Espinoza, Jose Luis Vazquez; Liniger, Alexander; Schwarting, Wilko; Rus, Daniela, and Van Gool, Luc. Deep interactive motion prediction and planning: Playing games with motion prediction models. In *Learning for Dynamics and Control Conference*, pages 1006–1019. PMLR, 2022. (Cited on page 15.)

Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J., and Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Intl. Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010. URL `https://pdfs.semanticscholar.org/0ee1/916a0cb2dc7d3add086b5f1092c3d4beb38a.pdf`. (Cited on page 55.)

Everingham, Mark; Eslami, SM Ali; Van Gool, Luc; Williams, Christopher KI; Winn, John, and Zisserman, Andrew. The pascal visual object classes challenge: A retrospective. *Intl. Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015. (Cited on pages 30, 41 and 75.)

Fan, Lue; Xiong, Xuan; Wang, Feng; Wang, Naiyan, and Zhang, Zhaoxiang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2918–2927, 2021. (Cited on pages vii and 28.)

Fang, Liangji; Jiang, Qinhong; Shi, Jianping, and Zhou, Bolei. Tpnet: Trajectory proposal network for motion prediction. In *CVPR*, pages 6797–6806, 2020. (Cited on page 99.)

Gao, Jiyang; Sun, Chen; Zhao, Hang; Shen, Yi; Anguelov, Dragomir; Li, Congcong, and Schmid, Cordelia. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. (Cited on pages vii, 31 and 108.)

Gao, Peng; Ma, Teli; Li, Hongsheng; Lin, Ziyi; Dai, Jifeng, and Qiao, Yu. Mcmae: Masked convolution meets masked autoencoders. In *NeurIPS*, 2022. (Cited on pages 82 and 84.)

Garbade, Martin; Chen, Yueh-Tung; Sawatzky, Johann, and Gall, Juergen. Two stream 3d semantic scene completion. 2019. (Cited on page 15.)

Gehring, Jonas; Auli, Michael; Grangier, David; Yarats, Denis, and Dauphin, Yann N. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017. (Cited on page 25.)

Geiger, A.; Lenz, P., and Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012a. URL `http://www.cvlibs.net/publications/Geiger2012CVPR.pdf`. (Cited on pages 55 and 59.)

Geiger, Andreas; Lenz, Philip, and Urtasun, Raquel. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012b. (Cited on pages 29, 41 and 74.)

Gilles, Thomas; Sabatini, Stefano; Tsishkou, Dzmitry; Stanciulescu, Bogdan, and Moutarde, Fabien. Home: Heatmap output for future motion estimation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 500–507. IEEE, 2021a. (Cited on page 15.)

Gilles, Thomas; Sabatini, Stefano; Tsishkou, Dzmitry; Stanciulescu, Bogdan, and Moutarde, Fabien. Thomas: Trajectory heatmap output with learned multi-agent sampling. *arXiv preprint arXiv:2110.06607*, 2021b. (Cited on page 15.)

Gilles, Thomas; Sabatini, Stefano; Tsishkou, Dzmitry; Stanciulescu, Bogdan, and Moutarde, Fabien. Gohome: Graph-oriented heatmap output for future motion estimation. In *2022 international conference on robotics and automation (ICRA)*, pages 9107–9114. IEEE, 2022. (Cited on pages 15, 108, 114 and 117.)

Girgis, Roger; Golemo, Florian; Codevilla, Felipe; Weiss, Martin; D'Souza, Jim Aldon; Kahou, Samira Ebrahimi; Heide, Felix, and Pal, Christopher. Latent variable sequential set transformers for joint multi-agent motion prediction. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*. (Cited on page 114.)

Graham, Benjamin; Engelcke, Martin, and Van Der Maaten, Laurens. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. (Cited on pages vii and 27.)

Gu, Tianpei; Chen, Guangyi; Li, Junlong; Lin, Chunze; Rao, Yongming; Zhou, Jie, and Lu, Jiwen. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17113–17122, 2022. (Cited on pages 14, 99 and 102.)

Gupta, Agrim; Johnson, Justin; Fei-Fei, Li; Savarese, Silvio, and Alahi, Alexandre. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, pages 2255–2264, 2018. (Cited on pages xiv, 14, 32, 95, 99, 101 and 103.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 770–778, 2016a. (Cited on pages 22 and 37.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016b. (Cited on page 84.)

Hou, Yuenan; Zhu, Xinge; Ma, Yuexin; Loy, Chen Change, and Li, Yikang. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8479–8488, 2022. (Cited on page 12.)

Howard, Andrew; Sandler, Mark; Chu, Grace; Chen, Liang-Chieh; Chen, Bo; Tan, Mingxing; Wang, Weijun; Zhu, Yukun; Pang, Ruoming; Vasudevan, Vijay, and others, . Searching for MobileNetV3. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 1314–1324, 2019. (Cited on page 34.)

Howard, Andrew G; Zhu, Menglong; Chen, Bo; Kalenichenko, Dmitry; Wang, Weijun; Weyand, Tobias; Andreetto, Marco, and Adam, Hartwig. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. (Cited on page 37.)

Hu, Anthony; Murez, Zak; Mohan, Nikhil; Dudas, Sofía; Hawke, Jeffrey; Badrinarayanan, Vijay; Cipolla, Roberto, and Kendall, Alex. Fiery: future instance prediction in bird's-eye view from surround monocular cameras. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2021a. (Cited on page 15.)

Hu, Peiyun; Huang, Aaron; Dolan, John; Held, David, and Ramanan, Deva. Safe local motion planning with self-supervised freespace forecasting. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 12732–12741, 2021b. (Cited on page 15.)

Hu, Qingyong; Yang, Bo; Xie, Linhai; Rosa, Stefano; Guo, Yulan; Wang, Zhihua; Trigoni, Niki, and Markham, Andrew. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. *arXiv preprint arXiv:1911.11236*, 2019. (Cited on pages 43 and 47.)

Hu, Qingyong; Yang, Bo; Xie, Linhai; Rosa, Stefano; Guo, Yulan; Wang, Zhihua; Trigoni, Niki, and Markham, Andrew. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020a. (Cited on page 10.)

Hu, Qingyong; Yang, Bo; Xie, Linhai; Rosa, Stefano; Guo, Yulan; Wang, Zhihua; Trigoni, Niki, and Markham, Andrew. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020b. (Cited on page 85.)

Hu, Shengchao; Chen, Li; Wu, Penghao; Li, Hongyang; Yan, Junchi, and Tao, Dacheng. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *Proc. of the Europ. Conf. on Computer Vision*, pages 533–549. Springer, 2022a. (Cited on page 15.)

Hu, Yihan; Shao, Wenxin; Jiang, Bo; Chen, Jiajie; Chai, Siqi; Yang, Zhening; Qian, Jingyu; Zhou, Helong, and Liu, Qiang. Hope: Hierarchical spatial-temporal network for occupancy flow prediction. *arXiv preprint arXiv:2206.10118*, 2022b. (Cited on page 15.)

Huang, Yingfan; Bi, Huikun; Li, Zhaoxin; Mao, Tianlu, and Wang, Zhaoqi. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 6272–6281, 2019. (Cited on pages 13, 14, 99, 100, 102, 103, 104 and 105.)

Huang, Yuanhui; Zheng, Wenzhao; Zhang, Yunpeng; Zhou, Jie, and Lu, Jiwen. Tri-perspective view for vision-based 3d semantic occupancy prediction. *arXiv preprint arXiv:2302.07817*, 2023a. (Cited on page 15.)

Huang, Zhiyu; Mo, Xiaoyu, and Lv, Chen. Multi-modal motion prediction with transformer-based neural network for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2605–2611. IEEE, 2022a. (Cited on page 15.)

Huang, Zhiyu; Mo, Xiaoyu, and Lv, Chen. Recoat: A deep learning-based framework for multi-modal motion prediction in autonomous driving application. In *2022 IEEE 25th International*

Conference on Intelligent Transportation Systems (ITSC), pages 988–993. IEEE, 2022b. (Cited on page 15.)

Huang, Zhiyu; Liu, Haochen; Wu, Jingda, and Lv, Chen. Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2023b. (Cited on page 15.)

Jia, Xiaosong; Sun, Liting; Zhao, Hang; Tomizuka, Masayoshi, and Zhan, Wei. Multi-agent trajectory prediction by combining egocentric and allocentric views. In *Conference on Robot Learning*, pages 1434–1443. PMLR, 2022. (Cited on page 15.)

Kamousi, Pegah; Lazard, Sylvain; Maheshwari, Anil, and Wuhrer, Stefanie. Analysis of farthest point sampling for approximating geodesics in a graph. *Computational Geometry*, 57:1–7, 2016. (Cited on page 67.)

Khandelwal, Siddhesh; Qi, William; Singh, Jagjeet; Hartnett, Andrew, and Ramanan, Deva. What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*, 2020. (Cited on pages 114 and 117.)

Khurana, Tarasha; Hu, Peiyun; Dave, Achal; Ziglar, Jason; Held, David, and Ramanan, Deva. Differentiable raycasting for self-supervised occupancy forecasting. In *Proc. of the Europ. Conf. on Computer Vision*, pages 353–369. Springer, 2022. (Cited on page 15.)

Kim, ByeoungDo; Park, Seong Hyeon; Lee, Seokhwan; Khoshimjonov, Elbek; Kum, Dongsuk; Kim, Junsoo; Kim, Jeong Soo, and Choi, Jun Won. Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14636–14645, 2021. (Cited on pages 108, 109, 114 and 116.)

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 21.)

Kirillov, Alexander; Levinkov, Evgeny; Andres, Bjoern; Savchynskyy, Bogdan, and Rother, Carsten. InstanceCut: from edges to instances with MultiCut. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017. (Cited on page 40.)

Kirillov, Alexander; Mintun, Eric; Ravi, Nikhila; Mao, Hanzi; Rolland, Chloe; Gustafson, Laura; Xiao, Tete; Whitehead, Spencer; Berg, Alexander C.; Lo, Wan-Yen; Dollár, Piotr, and Girshick, Ross. Segment anything. *arXiv:2304.02643*, 2023. (Cited on page 123.)

Kochanov, Deyvid; Nejadasl, Fatemeh Karimi, and Booij, Olaf. Kprnet: Improving projection-based lidar semantic segmentation. *arXiv preprint arXiv:2007.12668*, 2020a. (Cited on page 12.)

Kochanov, Deyvid; Nejadasl, Fatemeh Karimi, and Booij, Olaf. Kprnet: Improving projection-based lidar semantic segmentation. In *arXiv*, 2020b. (Cited on pages 13 and 85.)

Kong, Lingdong; Liu, Youquan; Chen, Runnan; Ma, Yuexin; Zhu, Xinge; Li, Yikang; Hou, Yuenan; Qiao, Yu, and Liu, Ziwei. Rethinking range view representation for lidar segmentation. *arXiv preprint arXiv:2303.05367*, 2023a. (Cited on page 12.)

Kong, Lingdong; Liu, Youquan; Chen, Runnan; Ma, Yuexin; Zhu, Xinge; Li, Yikang; Hou, Yuenan; Qiao, Yu, and Liu, Ziwei. Rethinking range view representation for lidar segmentation. In *ICCV*, 2023b. (Cited on pages 13 and 86.)

Kosaraju, Vineet; Sadeghian, Amir; Martín-Martín, Roberto; Reid, Ian; Rezatofighi, Hamid, and Savarese, Silvio. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *NIPS*, pages 137–146, 2019. (Cited on pages 14, 95, 96, 99 and 102.)

Krizhevsky, Alex; Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. (Cited on page 22.)

Lai, Xin; Chen, Yukang; Lu, Fanbin; Liu, Jianhui, and Jia, Jiaya. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17545–17555, 2023. (Cited on page 11.)

Landrieu, Loic and Simonovsky, Martin. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. (Cited on pages 10, 43, 75 and 78.)

Lang, Alex H; Vora, Sourabh; Caesar, Holger; Zhou, Lubing; Yang, Jiong, and Beijbom, Oscar. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. (Cited on page 27.)

LeCun, Yann; Bottou, Léon; Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Cited on pages vii and 22.)

Lee, Chen-Yu; Xie, Saining; Gallagher, Patrick; Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015. (Cited on page 40.)

Lerner, Alon; Chrysanthou, Yiorgos, and Lischinski, Dani. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007. (Cited on pages xiii, xiv, 32, 99, 102 and 104.)

Li, Hanchao; Xiong, Pengfei; An, Jie, and Wang, Lingxue. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018. (Cited on pages 43 and 75.)

Li, Jiachen; Ma, Hengbo, and Tomizuka, Masayoshi. Conditional generative neural system for probabilistic trajectory prediction. *arXiv preprint arXiv:1905.01631*, 2019. (Cited on pages 14, 95 and 99.)

Li, Kunchang; Wang, Yali; Gao, Peng; Song, Guanglu; Liu, Yu; Li, Hongsheng, and Qiao, Yu. Uniformer: Unified transformer for efficient spatiotemporal representation learning. In *arXiv*, 2022. (Cited on pages 82 and 84.)

Li, Shi-Jie; AbuFarha, Yazan; Liu, Yun; Cheng, Ming-Ming, and Gall, Juergen. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2020a. (Cited on page 97.)

Li, Shijie; Chen, Xieyuanli; Liu, Yun; Dai, Dengxin; Stachniss, Cyrill, and Gall, Juergen. Multi-scale interaction for real-time lidar data segmentation on an embedded platform. *arXiv preprint arXiv:2008.09162*, 2020b. (Cited on pages 51, 54 and 55.)

Li, Shijie; Liu, Yun, and Gall, Juergen. Projected-point-based segmentation: A new paradigm for LiDAR point cloud segmentation. *arXiv preprint arXiv:2008.03928*, 2020c. (Cited on pages 43 and 47.)

Li, Shijie; Chen, Xieyuanli; Liu, Yun; Dai, Dengxin; Stachniss, Cyrill, and Gall, Juergen. Multi-scale interaction for real-time lidar data segmentation on an embedded platform. In *R-AL*. IEEE, 2021a. (Cited on pages 7, 86 and 87.)

Li, Shijie; Liu, Yun, and Gall, Juergen. Rethinking 3-d lidar point cloud segmentation. In *IEEE Transactions on Neural Networks and Learning Systems*, 2021b. (Cited on page 7.)

Li, Shijie; Zhou, Yanying; Yi, Jinhui, and Gall, Juergen. Spatial-temporal consistency network for low-latency trajectory forecasting. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 1940–1949, 2021c. (Cited on page 7.)

Liang, Junwei; Jiang, Lu; Niebles, Juan Carlos; Hauptmann, Alexander G, and Fei-Fei, Li. Peeking into the future: Predicting future person activities and locations in videos. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 5725–5734, 2019. (Cited on pages xiv, 14, 95, 99 and 103.)

Liang, Junwei; Jiang, Lu, and Hauptmann, Alexander. Simaug: Learning robust representations from 3d simulation for pedestrian trajectory prediction in unseen cameras. *arXiv preprint arXiv:2004.02022*, 2020a. (Cited on pages 101, 102 and 103.)

Liang, Ming; Yang, Bin; Hu, Rui; Chen, Yun; Liao, Renjie; Feng, Song, and Urtasun, Raquel. Learning lane graph representations for motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 541–556. Springer, 2020b. (Cited on pages 114 and 115.)

Lin, Tsung-Yi; Goyal, Priya; Girshick, Ross; He, Kaiming, and Dollár, Piotr. Focal loss for dense object detection. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 2980–2988, 2017. (Cited on page 45.)

Liu, Jiahui; Chang, Chirui; Liu, Jianhui; Wu, Xiaoyang; Ma, Lan, and Qi, Xiaojuan. Mars3d: A plug-and-play motion-aware model for semantic segmentation on multi-scan 3d point clouds. In *CVPR*, 2023a. (Cited on pages 13, 87 and 88.)

Liu, Xingyu; Qi, Charles R, and Guibas, Leonidas J. Flownet3d: Learning scene flow in 3d point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2019a. (Cited on page 57.)

Liu, Youquan; Kong, Lingdong; Cen, Jun; Chen, Runnan; Zhang, Wenwei; Pan, Liang; Chen, Kai, and Liu, Ziwei. Segment any point cloud sequences by distilling vision foundation models. In *arXiv*, 2023b. (Cited on page 13.)

Liu, Yun; Cheng, Ming-Ming; Fan, Deng-Ping; Zhang, Le; Bian, JiaWang, and Tao, Dacheng. Semantic edge detection with diverse deep supervision. *arXiv preprint arXiv:1804.02864*, 2018. (Cited on pages 35 and 40.)

Liu, Yun; Cheng, Ming-Ming; Hu, Xiaowei; Bian, Jia-Wang; Zhang, Le; Bai, Xiang, and Tang, Jinhui. Richer convolutional features for edge detection. *IEEE Trans. on Pattern Analalysis and Machine Intelligence*, 41(8):1939–1946, 2019b. (Cited on pages 35 and 40.)

Liu, Zhijian; Tang, Haotian; Lin, Yujun, and Han, Song. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, 2019c. (Cited on pages 87, 88 and 89.)

Luo, Chenxu; Sun, Lin; Dabiri, Dariush, and Yuille, Alan. Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 2370–2376. IEEE, 2020a. (Cited on page 114.)

Luo, Xiaotong; Xie, Yuan; Zhang, Yulun; Qu, Yanyun; Li, Cuihua, and Fu, Yun. Latticenet: Towards lightweight image super-resolution with lattice block. In *ECCV*, 2020b. (Cited on page 85.)

Ma, Junyi; Chen, Xieyuanli; Xu, Jingyi, and Xiong, Guangming. Seqot: A spatial-temporal transformer network for place recognition using sequential lidar data. In *IEEE Transactions on Industrial Electronics*, 2022a. (Cited on page 13.)

Ma, Junyi; Zhang, Jun; Xu, Jintao; Ai, Rui; Gu, Weihao, and Chen, Xieyuanli. Overlaptransformer: An efficient and yaw-angle-invariant transformer network for lidar-based place recognition. In *R-AL*, 2022b. (Cited on page 13.)

Ma, Ningning; Zhang, Xiangyu; Zheng, Hai-Tao, and Sun, Jian. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proc. of the Europ. Conf. on Computer Vision*, pages 116–131, 2018. (Cited on page 34.)

Ma, Teli; Wang, Mengmeng; Xiao, Jimin; Wu, Huifeng, and Liu, Yong. Synchronize feature extracting and matching: A single branch framework for 3d object tracking. *arXiv*, ICCV. (Cited on page 84.)

Mangalam, Karttikeya; Girase, Harshayu; Agarwal, Shreyas; Lee, Kuan-Hui; Adeli, Ehsan; Malik, Jitendra, and Gaidon, Adrien. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Proc. of the Europ. Conf. on Computer Vision*, pages 759–776. Springer, 2020. (Cited on pages 14, 99, 101, 102 and 103.)

Mangalam, Karttikeya; An, Yang; Girase, Harshayu, and Malik, Jitendra. From goals, waypoints & paths to long term human trajectory forecasting. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 15233–15242, 2021. (Cited on page 14.)

Mao, Weibo; Xu, Chenxin; Zhu, Qi; Chen, Siheng, and Wang, Yanfeng. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5517–5526, 2023. (Cited on pages 99 and 102.)

Messaoud, Kaouther; Deo, Nachiket; Trivedi, Mohan M, and Nashashibi, Fawzi. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 165–170. IEEE, 2021. (Cited on page 114.)

Mildenhall, Ben; Srinivasan, Pratul P; Tancik, Matthew; Barron, Jonathan T; Ramamoorthi, Ravi, and Ng, Ren. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. (Cited on page 123.)

Milioto, Andres; Vizzo, Ignacio; Behley, Jens, and Stachniss, Cyrill. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4213–4220. IEEE, 2019a. (Cited on pages 54, 55 and 56.)

Milioto, Andres; Vizzo, Ignacio; Behley, Jens, and Stachniss, Cyrill. RangeNet++: Fast and accurate LiDAR semantic segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 4213–4220, 2019b. (Cited on pages 11, 34, 35, 38, 40, 43, 45, 46, 47, 48, 51, 64, 71, 75, 76, 78 and 79.)

Milioto, Andres; Vizzo, Ignacio; Behley, Jens, and Stachniss, Cyrill. Rangenet++: Fast and accurate lidar semantic segmentation. In *IROS*, 2019c. (Cited on pages x, 13, 81, 82, 83, 86, 87, 88, 89 and 90.)

Mo, Xiaoyu; Huang, Zhiyu; Xing, Yang, and Lv, Chen. Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9554–9567, 2022. (Cited on page 15.)

Mohamed, Abduallah; Qian, Kun; Elhoseiny, Mohamed, and Claudel, Christian. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020. (Cited on pages x, xiv, 13, 14, 94, 95, 96, 99, 100, 101, 102, 103, 104 and 105.)

Mohapatra, Sambit; Hodaei, Mona; Yogamani, Senthil; Milz, Stefan; Gotzig, Heinrich; Simon, Martin; Rashed, Hazem, and Maeder, Patrick. Limoseg: Real-time bird's eye view based lidar motion segmentation. *arXiv preprint arXiv:2111.04875*, 2021. (Cited on page 27.)

Monti, Alessio; Bertugli, Alessia; Calderara, Simone, and Cucchiara, Rita. Dag-net: Double attentive graph neural network for trajectory forecasting. *arXiv preprint arXiv:2005.12661*, 2020. (Cited on pages xiv, 14, 95, 99, 100, 102, 103 and 104.)

Müller, Thomas; Evans, Alex; Schied, Christoph, and Keller, Alexander. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15, 2022. (Cited on page 123.)

Narayanan, Sriram; Moslemi, Ramin; Pittaluga, Francesco; Liu, Buyu, and Chandraker, Manmohan. Divide-and-conquer for lane-aware diverse trajectory prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 15799–15808, 2021. (Cited on page 114.)

Nayakanti, Nigamaa; Al-Rfou, Rami; Zhou, Aurick; Goel, Kratarth; Refaat, Khaled S, and Sapp, Benjamin. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2980–2987. IEEE, 2023. (Cited on page 15.)

Ngiam, Jiquan; Vasudevan, Vijay; Caine, Benjamin; Zhang, Zhengdong; Chiang, Hao-Tien Lewis; Ling, Jeffrey; Roelofs, Rebecca; Bewley, Alex; Liu, Chenxi; Venugopal, Ashish, and others, . Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*, 2021. (Cited on page 15.)

Pan, Yancheng; Gao, Biao; Mei, Jilin; Geng, Sibo; Li, Chengkun, and Zhao, Huijing. Semanticposs: A point cloud dataset with large quantity of dynamic instances. *arXiv preprint arXiv:2002.09147*, 2020a. (Cited on pages 29, 41, 47 and 74.)

Pan, Yancheng; Gao, Biao; Mei, Jilin; Geng, Sibo; Li, Chengkun, and Zhao, Huijing. Semanticposs: A point cloud dataset with large quantity of dynamic instances. In *IV*, 2020b. (Cited on pages 83 and 86.)

Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca, and others, . PyTorch: An imperative style, high-performance deep learning library. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pages 8024–8035, 2019. (Cited on pages ix and 72.)

Pellegrini, Stefano; Ess, Andreas; Schindler, Konrad, and Van Gool, Luc. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, pages 261–268. IEEE, 2009. (Cited on pages xiii, xiv, 32, 99, 102 and 104.)

Phan-Minh, Tung; Grigore, Elena Corina; Boulton, Freddy A; Beijbom, Oscar, and Wolff, Eric M. Covernet: Multimodal behavior prediction using trajectory sets. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. (Cited on pages xiv, 108, 114, 115 and 117.)

Pourheydari, Saber; Bahrami, Emad; Fayyaz, Mohsen; Francesca, Gianpiero; Noroozi, Mehdi; Gall, Juergen, and Samsung, AI. Taylorswiftnet: Taylor driven temporal modeling for swift future frame prediction. (Cited on page 123.)

Prakash, Aditya; Chitta, Kashyap, and Geiger, Andreas. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021. (Cited on page 15.)

Pumarola, Albert; Corona, Enric; Pons-Moll, Gerard, and Moreno-Noguer, Francesc. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021. (Cited on page 123.)

Qi, Charles R; Su, Hao; Mo, Kaichun, and Guibas, Leonidas J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017a. (Cited on page 85.)

Qi, Charles R; Su, Hao; Mo, Kaichun, and Guibas, Leonidas J. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 652–660, 2017b. (Cited on pages 10, 34, 43, 75 and 78.)

Qi, Charles Ruizhongtai; Yi, Li; Su, Hao, and Guibas, Leonidas J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017c. (Cited on page 85.)

Qi, Charles Ruizhongtai; Yi, Li; Su, Hao, and Guibas, Leonidas J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017d. (Cited on pages ix, 10, 34, 43, 64, 65, 66, 67, 68, 69, 72, 73, 75, 76, 78 and 108.)

Qiu, Haibo; Yu, Baosheng, and Tao, Dacheng. Gfnet: Geometric flow network for 3d point cloud semantic segmentation. *arXiv preprint arXiv:2207.02605*, 2022. (Cited on page 12.)

Qiu, Shi; Anwar, Saeed, and Barnes, Nick. Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *CVPR*, 2021. (Cited on page 85.)

Razani, Ryan; Cheng, Ran; Taghavi, Ehsan, and Bingbing, Liu. Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions. In *ICRA*, 2021. (Cited on pages 85 and 87.)

Robicquet, Alexandre; Sadeghian, Amir; Alahi, Alexandre, and Savarese, Silvio. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, pages 549–565. Springer, 2016. (Cited on pages xiii, xiv, 32, 99, 100, 101, 102 and 104.)

Ronneberger, Olaf; Fischer, Philipp, and Brox, Thomas. U-Net: Convolutional networks for biomedical image segmentation. pages 234–241, 2015. (Cited on page 34.)

Rosu, Radu Alexandru; Schütt, Peer; Quenzel, Jan, and Behnke, Sven. LatticeNet: Fast point cloud segmentation using permutohedral lattices. *arXiv preprint arXiv:1912.05905*, 2019. (Cited on page 10.)

Sadeghian, Amir; Kosaraju, Vineet; Sadeghian, Ali; Hirose, Noriaki; Rezatofighi, Hamid, and Savarese, Silvio. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *CVPR*, pages 1349–1358, 2019. (Cited on pages 14, 95, 99 and 101.)

Salzmann, Tim; Ivanovic, Boris; Chakravarty, Punarjay, and Pavone, Marco. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020. (Cited on pages xiv, 15, 108, 114, 115 and 117.)

Sandler, Mark; Howard, Andrew; Zhu, Menglong; Zhmoginov, Andrey, and Chen, Liang-Chieh. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. (Cited on page 37.)

Shi, Hanyu; Lin, Guosheng; Wang, Hao; Hung, Tzu-Yi, and Wang, Zhenhua. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020a. (Cited on page 57.)

Shi, Shaoshuai; Guo, Chaoxu; Jiang, Li; Wang, Zhe; Shi, Jianping; Wang, Xiaogang, and Li, Hongsheng. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020b. (Cited on page 79.)

Shi, Shaoshuai; Jiang, Li; Dai, Dengxin, and Schiele, Bernt. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543, 2022. (Cited on page 15.)

Shi, Yining; Jiang, Kun; Wang, Ke; Li, Jiusi; Wang, Yunlong, and Yang, Diange. Fusionmotion: Multi-sensor asynchronous fusion for continuous occupancy prediction via neural-ode. *arXiv preprint arXiv:2302.09585*, 2023. (Cited on page 15.)

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 22.)

Song, Haoran; Ding, Wenchao; Chen, Yuxuan; Shen, Shaojie; Wang, Michael Yu, and Chen, Qifeng. Pip: Planning-informed trajectory prediction for autonomous driving. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 598–614. Springer, 2020. (Cited on page 15.)

Su, Hang; Jampani, Varun; Sun, Deqing; Maji, Subhransu; Kalogerakis, Evangelos; Yang, Ming-Hsuan, and Kautz, Jan. SPLATNet: Sparse lattice networks for point cloud processing. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. (Cited on pages 10, 43, 75 and 78.)

Sun, Jiadai; Dai, Yuchao; Zhang, Xianjing; Xu, Jintao; Ai, Rui; Gu, Weihao, and Chen, Xieyuanli. Efficient spatial-temporal information fusion for lidar-based 3d moving object segmentation. In *IROS*, 2022a. (Cited on pages 13, 87 and 88.)

Sun, Jianhua; Jiang, Qinhong, and Lu, Cewu. Recursive social behavior graph for trajectory prediction. In *CVPR*, pages 660–669, 2020. (Cited on pages 14, 99 and 102.)

Sun, Jianhua; Li, Yuxuan; Fang, Hao-Shu, and Lu, Cewu. Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13250–13259, 2021. (Cited on page 14.)

Sun, Qiao; Huang, Xin; Williams, Brian C, and Zhao, Hang. Intersim: Interactive traffic simulation via explicit relation modeling. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11416–11423. IEEE, 2022b. (Cited on page 15.)

Tang, Haotian; Liu, Zhijian; Zhao, Shengyu; Lin, Yujun; Lin, Ji; Wang, Hanrui, and Han, Song. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. (Cited on page 10.)

Tatarchenko, Maxim; Park, Jaesik; Koltun, Vladlen, and Zhou, Qian-Yi. Tangent convolutions for dense prediction in 3D. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. (Cited on pages 10, 43, 75 and 78.)

Thomas, H.; Qi, C.R.; Deschaud, J.; Marcotegui, B.; Goulette, F., and Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2019a. URL `proceedings:thomas2019iccv.pdf`. (Cited on page 57.)

Thomas, Hugues; Qi, Charles R; Deschaud, Jean-Emmanuel; Marcotegui, Beatriz; Goulette, François, and Guibas, Leonidas J. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019b. (Cited on page 85.)

Thomas, Hugues; Qi, Charles R; Deschaud, Jean-Emmanuel; Marcotegui, Beatriz; Goulette, François, and Guibas, Leonidas J. KPConv: Flexible and deformable convolution for point clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 6411–6420, 2019c. (Cited on page 10.)

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *NeurIPS*, 2017a. (Cited on pages 13, 82 and 84.)

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. *Advances in neural information processing systems*, 30, 2017b. (Cited on pages vii, 12, 23, 24 and 112.)

Wang, Chuhua; Wang, Yuchen; Xu, Mingze, and Crandall, David J. Stepwise goal-driven networks for trajectory prediction. *IEEE Robotics and Automation Letters*, 7(2):2716–2723, 2022a. (Cited on pages 14 and 114.)

Wang, Hao; Wang, Weining, and Liu, Jing. Temporal memory attention for video semantic segmentation. In *ICIP*, 2021. (Cited on pages 87 and 88.)

Wang, Limin; Xiong, Yuanjun; Wang, Zhe; Qiao, Yu; Lin, Dahua; Tang, Xiaoou, and Van Gool, Luc. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018. (Cited on page 53.)

Wang, Mengmeng; Ma, Teli; Zuo, Xingxing; Lv, Jiajun, and Liu, Yong. Correlation pyramid network for 3d single object tracking. In *CVPR*, 2023. (Cited on page 84.)

Wang, Song; Zhu, Jianke, and Zhang, Ruixiang. Meta-rangeseg: Lidar sequence semantic segmentation using multiple feature aggregation. In *R-AL*, 2022b. (Cited on pages 13, 85, 88 and 89.)

Williams, Travis and Li, Robert. Wavelet pooling for convolutional neural networks. In *ICLR*, 2018. (Cited on pages 95 and 96.)

Wu, Bichen; Wan, Alvin; Yue, Xiangyu, and Keutzer, Kurt. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, 2018a. (Cited on pages 13, 82, 83, 86, 87 and 88.)

Wu, Bichen; Wan, Alvin; Yue, Xiangyu, and Keutzer, Kurt. SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, pages 1887–1893, 2018b. (Cited on pages 11, 12, 34, 38, 43, 45, 47, 64, 67, 75, 76 and 78.)

Wu, Bichen; Zhou, Xuanyu; Zhao, Sicheng; Yue, Xiangyu, and Keutzer, Kurt. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019a. (Cited on pages 86 and 87.)

Wu, Bichen; Zhou, Xuanyu; Zhao, Sicheng; Yue, Xiangyu, and Keutzer, Kurt. SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, pages 4376–4382, 2019b. (Cited on pages 12, 34, 38, 43, 45, 47, 64, 75, 76 and 78.)

Wu, Wenxuan; Qi, Zhongang, and Fuxin, Li. PointConv: Deep convolutional networks on 3D point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019c. (Cited on pages ix, 34, 64, 65, 68, 69, 72, 73 and 76.)

Xia, Zhaoyang; Liu, Youquan; Li, Xin; Zhu, Xinge; Ma, Yuexin; Li, Yikang; Hou, Yuenan, and Qiao, Yu. Scpnet: Semantic scene completion on point cloud. In *CVPR*, 2023. (Cited on page 13.)

Xiao, Aoran; Yang, Xiaofei; Lu, Shijian; Guan, Dayan, and Huang, Jiaxing. Fps-net: A convolutional fusion network for large-scale lidar point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176:237–249, 2021. (Cited on page 12.)

Xie, Enze; Wang, Wenhai; Yu, Zhiding; Anandkumar, Anima; Alvarez, Jose M, and Luo, Ping. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. (Cited on page 84.)

Xu, Chenfeng; Wu, Bichen; Wang, Zining; Zhan, Wei; Vajda, Peter; Keutzer, Kurt, and Tomizuka, Masayoshi. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *ECCV*, 2020a. (Cited on page 87.)

Xu, Chenfeng; Wu, Bichen; Wang, Zining; Zhan, Wei; Vajda, Peter; Keutzer, Kurt, and Tomizuka, Masayoshi. SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation. *arXiv preprint arXiv:2004.01803*, 2020b. (Cited on page 12.)

Xu, Chenxin; Mao, Weibo; Zhang, Wenjun, and Chen, Siheng. Remember intentions: Retrospective-memory-based trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6488–6497, 2022. (Cited on pages 13, 14 and 99.)

Xu, Yifan; Fan, Tianqi; Xu, Mingye; Zeng, Long, and Qiao, Yu. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *European Conference on Computer Vision*, pages 87–102, 2018. (Cited on pages ix, 64, 65, 69, 72, 73 and 76.)

Yan, Xu; Gao, Jiantao; Li, Jie; Zhang, Ruimao; Li, Zhen; Huang, Rui, and Cui, Shuguang. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3101–3109, 2021. (Cited on page 11.)

Yan, Xu; Gao, Jiantao; Zheng, Chaoda; Zheng, Chao; Zhang, Ruimao; Cui, Shuguang, and Li, Zhen. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European Conference on Computer Vision*, pages 677–695. Springer, 2022. (Cited on page 12.)

Yang, Chenyu; Chen, Yuntao; Tian, Hao; Tao, Chenxin; Zhu, Xizhou; Zhang, Zhaoxiang; Huang, Gao; Li, Hongyang; Qiao, Yu; Lu, Lewei, and others, . Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In *CVPR*, 2023. (Cited on pages 87 and 88.)

Yang, Maoke; Yu, Kun; Zhang, Chi; Li, Zhiwei, and Yang, Kuiyuan. DenseASPP for semantic segmentation in street scenes. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 3684–3692, 2018. (Cited on pages 43, 47, 75, 76 and 78.)

Ye, Maosheng; Cao, Tongyi, and Chen, Qifeng. Tpcn: Temporal point cloud networks for motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11318–11327, 2021. (Cited on pages 108 and 110.)

Yin, Tianwei; Zhou, Xingyi, and Krahenbuhl, Philipp. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. (Cited on page 27.)

Yoon, David; Tang, Tim, and Barfoot, Timothy. Mapless online detection of dynamic objects in 3d lidar. In *Proc. of the Conf. on Computer and Robot Vision*, pages 113–120, 2019. (Cited on page 57.)

Yu, Changqian; Wang, Jingbo; Peng, Chao; Gao, Changxin; Yu, Gang, and Sang, Nong. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In *Proc. of the Europ. Conf. on Computer Vision*, pages 325–341, 2018a. (Cited on pages 43 and 47.)

Yu, Changqian; Wang, Jingbo; Peng, Chao; Gao, Changxin; Yu, Gang, and Sang, Nong. Learning a discriminative feature network for semantic segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 1857–1866, 2018b. (Cited on page 40.)

Yuan, Ye; Weng, Xinshuo; Ou, Yanglan, and Kitani, Kris M. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021. (Cited on pages xiv, 99, 108, 109, 114, 115 and 117.)

Yue, Jiangbei; Manocha, Dinesh, and Wang, He. Human trajectory prediction via neural social physics. In *Proc. of the Europ. Conf. on Computer Vision*, pages 376–394. Springer, 2022. (Cited on page 13.)

Zhang, Hang; Dana, Kristin; Shi, Jianping; Zhang, Zhongyue; Wang, Xiaogang; Tyagi, Ambrish, and Agrawal, Amit. Context encoding for semantic segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018. (Cited on pages 35 and 40.)

Zhang, Lidan; She, Qi, and Guo, Ping. Stochastic trajectory prediction with social graph network. *arXiv preprint arXiv:1907.10233*, 2019a. (Cited on pages 99 and 102.)

Zhang, Pu; Ouyang, Wanli; Zhang, Pengfei; Xue, Jianru, and Zheng, Nanning. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 12085–12094, 2019b. (Cited on pages 14, 95, 99 and 103.)

Zhang, Yang; Zhou, Zixiang; David, Philip; Yue, Xiangyu; Xi, Zerong, and Foroosh, Hassan. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *arXiv*, 2020a. (Cited on pages 88 and 89.)

Zhang, Yang; Zhou, Zixiang; David, Philip; Yue, Xiangyu; Xi, Zerong; Gong, Boqing, and Foroosh, Hassan. PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020b. (Cited on pages 11, 43 and 47.)

Zhang, Yunpeng; Zhu, Zheng; Zheng, Wenzhao; Huang, Junjie; Huang, Guan; Zhou, Jie, and Lu, Jiwen. Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving. *arXiv preprint arXiv:2205.09743*, 2022. (Cited on page 15.)

Zhao, Hengshuang; Shi, Jianping; Qi, Xiaojuan; Wang, Xiaogang, and Jia, Jiaya. Pyramid scene parsing network. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017. (Cited on pages 34, 35, 40, 47, 75, 76 and 78.)

Zhao, Yiming; Bai, Lin, and Huang, Xinming. Fidnet: Lidar point cloud semantic segmentation with fully interpolation decoding. In *IROS*, 2021. (Cited on pages 13, 81, 82, 85, 86, 87, 88 and 89.)

Zhi, Shuaifeng; Laidlow, Tristan; Leutenegger, Stefan, and Davison, Andrew J. In-place scene labelling and understanding with implicit scene representation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2021. (Cited on page 123.)

Zhou, Q.Y.; Park, J., and Koltun, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. URL `https://arxiv.org/pdf/1801.09847.pdf`. (Cited on page 59.)

Zhu, Xinge; Zhou, Hui; Wang, Tai; Hong, Fangzhou; Ma, Yuexin; Li, Wei; Li, Hongsheng, and Lin, Dahua. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9939–9948, 2021. (Cited on page 11.)